THESE

présentée à

L' UNIVERSITE PIERRE ET MARIE CURIE (PARIS VI)

pour l'obtention du

DOCTORAT TROISIEME CYCLE

Spécialité : MATHEMATIQUES

Mention : INFORMATIQUE

par M. Jean-Philippe RICHARD

Sujet de la thèse : Sur l'algorithme "Gauche-Droite" de planéarité des graphes, de MM. H. de FRAYSSEIX et P. ROSENSTIEHL.

Soutenue le 30 Juin 1982 devant la Commission

composée de : Président M. J.C. SIMON

Examinateurs : MM. M. CHEIN

H.de FRAYSSEIX

M. HABIB

P. ROSENSTIEHL

THE LIBRARY OF THE UNIVERSITY OF ILLINOIS

THESE

présentée à

L' UNIVERSITE PIERRE ET MARIE CURIE (PARIS VI)

pour l'obtention du

DOCTORAT TROISIEME CYCLE

Spécialité : MATHEMATIQUES

Mention : INFORMATIQUE

par M. Jean-Philippe RICHARD

Sujet de la thèse : Sur l'algorithme "Gauche-Droite" de planéarité des graphes, de MM. H. de FRAYSSEIX et P. ROSENSTIEHL.

Soutenue le 30 Juin 1982 devant la Commission

composée de : Président M. J.C. SIMON

Examinateurs : MM. M. CHEIN

H.de FRAYSSEIX

M. HABIB

P. ROSENSTIEHL

Je remercie Monsieur Jean-Claude SIMON (Professeur à l'Université Pierre et Marie CURIE , Paris) qui m'a fait l'honneur de présider le jury de cette thèse.

J'exprime toute ma gratitude à Messieurs Michel CHEIN (Professeur à l'Université des Sciences et Techniques du Languedoc , Montpellier) , Hubert de FRAYSSEIX (Chercheur au Collège de France , Paris) , et Pierre ROSENSTIEHL (Directeur d'Etudes à l'Ecole des Hautes Etudes en Sciences Sociales , Paris) qui ont bien voulu accepter de participer au jury.

Je suis profondément reconnaissant à Monsieur Michel HABIB (Assistant à l'Université Pierre et Marie CURIE , Paris) d'avoir agréablement dirigé mes recherches et du soutien constant qu'il m'a apporté.

Je remercie les membres du service de tirage pour l'impression de cette thèse.

Enfin , je remercie Monsieur Didier TALLOT pour son aide assidue et je rappelle que le programme présenté ici a été réalisé avec sa collaboration.

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41547321_0008

Cette thèse est consacrée à l'étude de l'algorithme "Gauche-Droite" de planéarité des graphes , de MM. H. de FRAYSSEIX et P. ROSENSTIEHL.

Nous étudions des graphes non orientés simples. [réf.B.1]

Soit $G = (X,E)$ avec $|X| = N$ et $|E| = M$.

Cet algorithme est linéaire (au sens des sommets) en temps de calcul et en encombrement mémoire , c'est à dire que ces deux valeurs peuvent s'exprimer sous la forme "$\alpha.N + \beta$" , où α et β sont deux constantes. [réf.H.S.]

Au premier chapitre , nous décrivons l'algorithme ainsi que les structures utilisées. Nous présentons également l'aspect géométrique sous-jacent de cet algorithme.

Le second chapitre est constitué du listing du programme (écrit en PL1) ainsi que de six exécutions de ce même programme sur six graphes différents. Le dessin associé à chaque graphe est représenté.

Le chapitre III est entiérement consacré à l'analyse de ce programme et à la démonstration de sa linéarité.

Au chapitre IV nous présentons la partie "implantation" de l'algorithme , partie présente dans le programme.

SOMMAIRE

INTRODUCTION 1

CHAPITRE I : DESCRIPTION DE L'ALGORITHME 7

I/ Algorithme "Gauche-Droite" 9

1) Algorithme en pidgin 9

2) Procédure "TREMAUX" 10

3) Procédure "SORTING" 12

4) Procédure "PLANAR" 13

II/ La "Twin Unit" (TWUN) 16

1) Généralités 16

2) Préordre dans la Twin Unit 18

3) Description des règles de manipulation de la TWUN 19

3-1) X-fusion 20

3-2) H-fusion 21

3-3) Bascule 22

a) bascule "Droite-Gauche" 22

b) bascule "Gauche-Droite" 23

III/ Description de "FLIPPING1" 24

1) Introduction 24

2) Parenthèse 24

3) Algorithme 25

4) Remarques 26

IV/ Description de "FUFLICHOI" 28

1) Introduction 28

2) Algorithme 28

V/ Interprétation géométrique 29

1) Généralités 29

2) Interprétation des mécanismes élémentaires 31

2-1) X-fusion 31

2-2) H-fusion 31

2-3) Bascule 31

3) Interprétation de la TWUN 32

4) Exemple 34

VI/ Description des "Merges" 40

1) Introduction 40

2) Retour sur l'algorithme 40

CHAPITRE II : LISTING DU PROGRAMME ET EXEMPLES 51

I/ Introduction 53

II/ Programme 55

III/ 6 exécutions 74

CHAPITRE III : ANALYSE DU PROGRAMME 93

I/ Introduction 95

II/ Analyse en place mémoire 95

1) Généralités 95

2) Procédures "constantes" 96

3) Procédures "variables" 96

3-1) FUFLICHOI 97

3-2) MERGETHIN 97

3-3) MERGETHICK 97

3-4) PLANAR 98

3-5) EMBED 98

4) Procédure principale 98

III/ Analyse en "temps" 100

1) Généralités 100

2) Calcul de T (TREMAUX) et T (SORTING) 102

3) Calcul de T (PLANAR) 102

3-1) T_G(X ou H-FUSION) 103

3-2) T_G(FLIPPING1,2,3) 104

3-3) T_G(FUFLICHOI) 105

3-4) T_G(élimination) 107

3-5) Fin de l'analyse de "PLANAR" 108

4) Calcul de T (EMBED) 109

5) Conclusions 110

CHAPITRE IV : IMPLANTATION ET PROLONGEMENTS 113

I/ Généralités 115

II/ La procédure "EMBED" 117

1) Introduction 117

2) Algorithme 118

BIBLIOGRAPHIE 123

~ INTRODUCTION ~

Rappelons qu'un graphe est 'planaire' s'il est possible de le représenter sur un plan de sorte que les sommets soient des points distincts , les arêtes des courbes simples , et que deux arêtes ne se rencontrent pas en dehors de leurs extrémités. [réf.B.1]

Les deux premiers graphes non planaires sont $K_{3,3}$ et K_5 .

Une 'face' d'un graphe planaire est par définition une région de plan limitée par des arêtes , et telle que deux points arbitraires dans cette région peuvent toujours être reliés par un trait continu ne rencontrant ni sommets , ni arêtes.

La formule d'Euler pour un graphe planaire connexe relie le nombre de faces 'F' aux nombres 'N' et 'M' : $N - M + F = 2$.

De cette relation connue depuis 1736 nous pouvons déduire un corollaire qui nous sera très utile : [réf.S.1]

Corollaire : Soit G un graphe simple planaire et connexe ,
alors : $M \leqslant 3.N - 6$.

Si nous devons écrire un algorithme qui reconnaisse si un graphe est planaire , cette condition nécessaire nous permet de rejeter tous les graphes dont le nombre d'arcs est strictement supérieur à $3.N - 6$.

Cette condition est également extrèmement importante pour la linéarité de l'algorithme : en effet , elle implique trivialement qu'un algorithme linéaire en "M" le sera aussi en "N". (ce qui est généralement faux car M est borné par N^2 pour un graphe quelconque).

Une première caractérisation des graphes planaires est le théorème de Kuratowski (1930): " Un graphe est planaire si et seulement s'il ne contient aucune subdivision de K_5 ou de $K_{3,3}$.

Cette caractérisation par sous-graphe exclu a eu une grande importance théorique.

Signalons la preuve combinatoire élégante de J.C. FOURNIER , [réf.F.1] .

Il y a eu d'autres caractérisations des graphes planaires , mais nous ne les mentionnerons pas ici; pour plus de détails se reporter par exemple au livre de C. BERGE [réf.B.1].

Pour le problème algorithmique de reconnaissance des graphes planaires qui nous concerne , le théorème de KURATOWSKI n'engendre pas un bon algorithme.

Signalons à ce sujet l'algorithme de P. MEI et N. GIBBS . [réf.M.G.] en $\mathcal{O}(N^6)$.

Cependant , dans les années 60 , apparurent les premiers algorithmes programmés.

Par exemple citons celui de G. DEMOUCRON , Y. MALGRANGE et R. PERTUISET. [réf.D.M.P.]

Citons également celui de A. LEMPEL , S. EVEN et I. CEDERBAUM. [réf.L.E.C.]

Remarquons que l'on ne se préocupait pas alors de la complexité de ces algorithmes.

Plus récemment (années 70) J. HOPCROFT et P. TARJAN proposèrent un algorithme en $\mathcal{O}(N \log N)$ [réf.H.T.1] puis en $\mathcal{O}(N)$ [réf.H.T.2] .

K.S. BOOTH et G.S. LEUKER ont également présenté un algorithme linéaire. [réf.B.L.]

Enfin , tout récemment , à Paris , H. de FRAYSSEIX et P. ROSENSTIEHL ont découvert des propriétés des graphes planaires autour de leurs arbres de Trémaux. [réf.F.R.1 , F.R.2 et F.R.3]

Ceci a permis l'élaboration de l'algorithme "Gauche-Droite" que nous avons programmé et analysé ci-après.

CHAPITRE I

DESCRIPTION DE L'ALGORITHME

I/ ALGORITHME "GAUCHE-DROITE"

1) ALGORITHME EN PIDGIN

1) Lire N et M
   Si $N \leqslant 4$ ou $M = N - 1$ alors G est planaire.
   Si $M > 3.N - 6$ alors G n'est pas planaire.

2) Lire la liste d'arêtes E et construire le graphe symétrique $\overleftrightarrow{G}$ associé.

3) A partir du sommet n°1 de $\overleftrightarrow{G}$, construire à l'aide d'un parcours de Trémaux (D.F.S.) un palmier P associé à G.

4) Classer les arcs de P suivant l'ordre décrit plus loin.

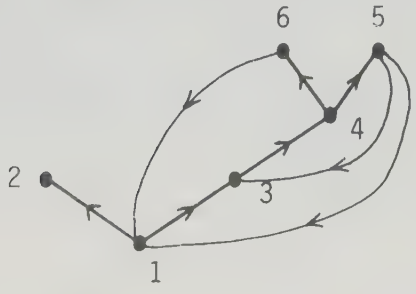
5) Pour chaque arc du coarbre du palmier P, vérifier sa compatibilité avec le préordre devant régner dans la Twin Unit.

L'étape 3) est réalisée par la procédure "TREMAUX", l'étape 4) par la procédure "SORTING" et enfin l'étape 5) par la procédure "PLANAR".

2) PROCEDURE "TREMAUX"

Définition: Un 'palmier' est composé d'une arborescence , des arcs de retour qui ferment les circuits , d'une 'bonne' numérotation des arcs et des sommets et enfin du calcul des LOW (≡ CHUT) et SLOW (≡CHUTP) des arcs.

Ex:



La procédure "TREMAUX" est un parcours en profondeur d'abord , (D.F.S.) [réf.R.N.D.] effectué sur le graphe symétrique $\overleftrightarrow{G}$.

Pendant ce parcours , il y a destruction du graphe symétrique $\overleftrightarrow{G}$ et construction d'un palmier.

Nous définissons ainsi l'arbre de Trémaux (*) et le coarbre.

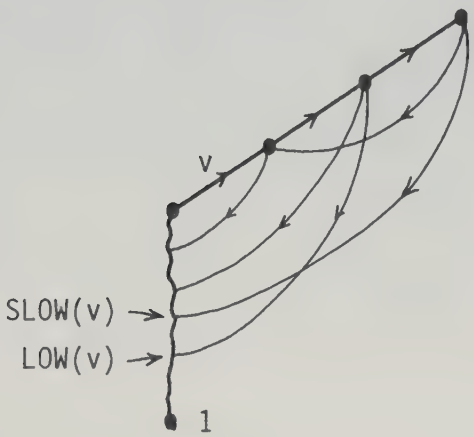
La renumérotation des arcs et des sommets est celle induite par le parcours. (avec une contrainte pour les arcs: les arcs de l'arbre sont numérotés de 1 à N-1 , et les arcs de coarbre de N à M .)

(*) E. Lucas , Récréations Mathématiques , Tome 1 , Blanchard , Paris 1921 .

Rappels sur LOW

Soit v = (x,y) un arc de l'arbre.

On définit le LOW(v) comme le numéro (nouveau) de sommet extrémité de l'arc de coarbre 'qui descend le plus bas' , le sommet origine de cet arc de coarbre appartenant au sous-arbre de racine y .



Par extension , on parle du LOW d'un arc de coarbre : il s'agit alors du numéro du sommet extrémité.

Définissons également le 'type' d'un arc d'arbre.

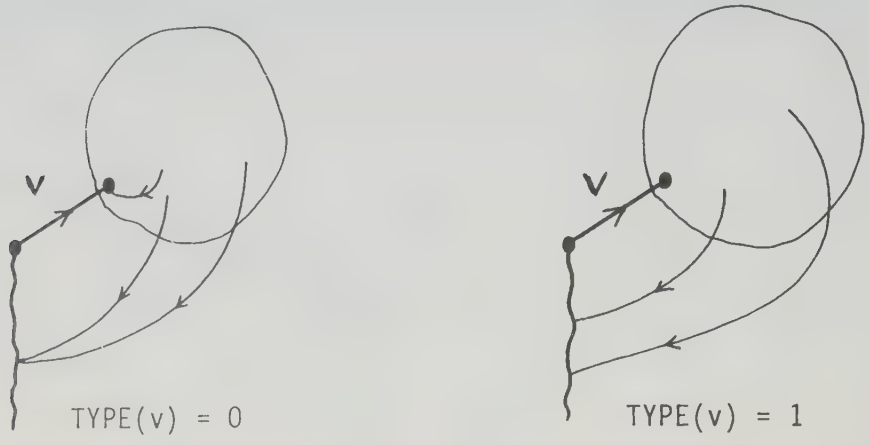
Un arc d'arbre est fin (thin) ou épais (thick).

Un arc v est fin ssi CHUTP(v) n'est pas modifié (depuis son initialisation) c-à-d. ssi CHUTP(v) = NUM(EX(v)) = NUM(EX(v))-1+1

——————— CHUTP(v) = v+1.

Sachant que CHUTP ne peut que décroitre , nous pouvons conclure : [v est fin] $\Leftrightarrow$ [CHUTP(v) $>$ v]

La variable TYPE(v) prendra la valeur 0 (resp. 1) si l'arc v est fin (resp. épais).



Le calcul du type des arcs de l'arbre est effectué à la fin de la procédure "TREMAUX" .

Nous rappelons qu'il est connu que le calcul d'un tel palmier est linéaire en $\mathcal{O}(\max(M,N))$.

3) PROCEDURE "SORTING"

Cette procédure de tri utilise le système Radix-sorting , dont la linéarité est bien connue. [réf.K.1]

Nous allons trier les arcs sortants , en chaque sommet.

Les arcs seront ensuite parcourus ("PLANAR") dans ce nouvel ordre.

Pour chaque sommet , nous allons différencier le premier arc de tous les autres.

Critère de choix pour le premier arc

C'est l'arc de l'arbre de plus petit LOW , à égalité fin a priorité sur épais.

Critère de tri pour les arcs suivants

Les arcs sortants restants sont triés par LOW croissant , à égalité coarbre a priorité sur arbre , et fin sur épais.

Rq: Dans le cas des sommets 'feuilles' de l'arbre , il n'existe pas de premier arc.

Nous construisons dans la procédure "SORTING" une nouvelle liste d'adjacence respectant cet ordre : SL (Sorting List)

4) PROCEDURE "PLANAR"

La procédure "PLANAR" est un nouveau parcours en profondeur d'abord (D.F.S.) construit sur la liste d'adjacence fournie par la procédure "SORTING" .

Il faut noter que nous retrouverons obligatoirement , lors de cette descente , le même arbre que celui obtenu par la procédure "TREMAUX" .

C'est au cours de ce dernier parcours en profondeur d'abord que la non-planéarité du graphe sera détectée. Si le parcours se termine normalement , alors le graphe sera déclaré 'planaire'.

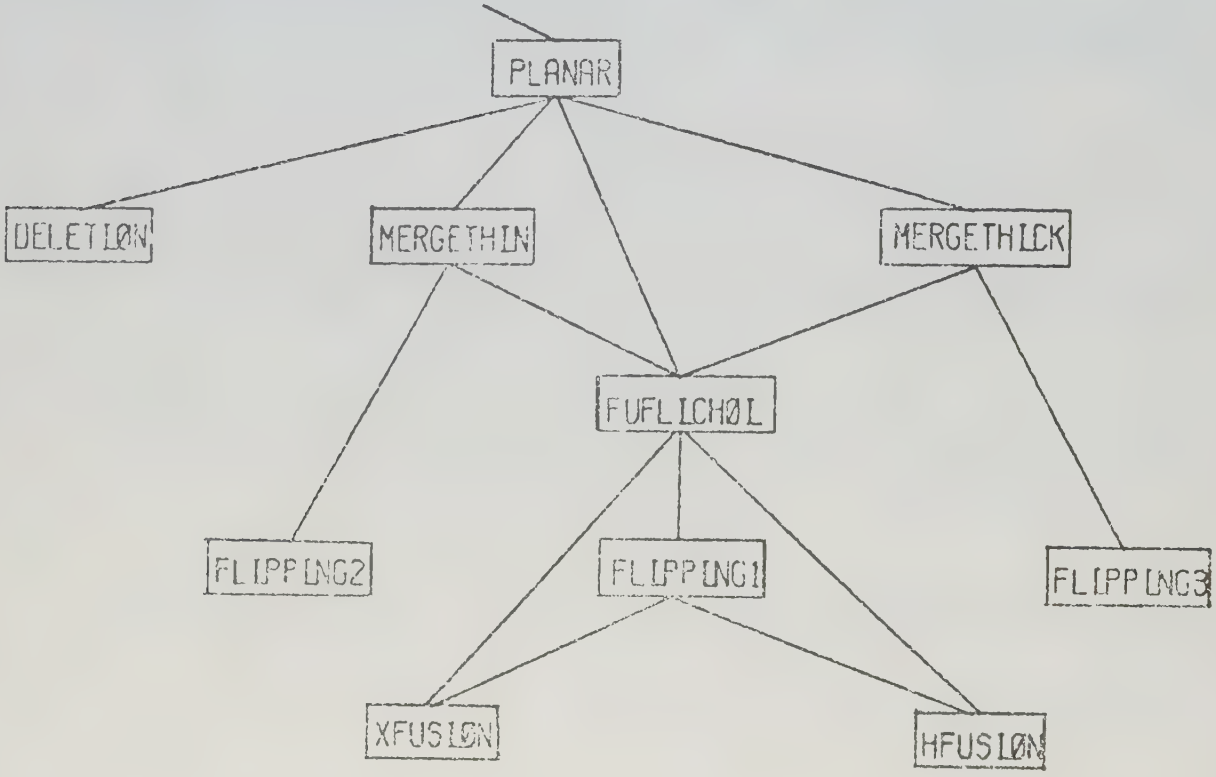


Schéma synoptique (partiel)

Les deux seules procédures où la non-planéarité du graphe peut être détectée sont "FLIPPING1" et "FLIPPING3".

Dans ce cas , via l'étiquette logique TILT , on revient immédiatement dans le programme principal "RIC".

Etape courante (soit $v = (x,y)$ un nouvel arc).

1) $\Leftrightarrow$ v est un arc de l'arbre.

- Si on a déjà fait un backtrack en x alors on ouvre dans la TWUN un plancher de branche de type KK.
- On passe à l'arc suivant (sortant de y).

2) $\Leftrightarrow$ v est un arc du coarbre.

- On cré un nouveau double élément dans la TWUN , on réorganise l'ordre dans la structure (call FUFLICHOI) et on passe à l'arc suivant (sortant de x).

Backtrack

Il n'existe plus d'arc nouveau sortant de x.

A-t-on fini (parcouru tous les arcs) ?

Sinon , on dépile dans la pile de backtrack le dernier arc d'arbre parcouru , et on détruit dans la TWUN tous les arcs (du co-arbre) ayant pour extrémité x.

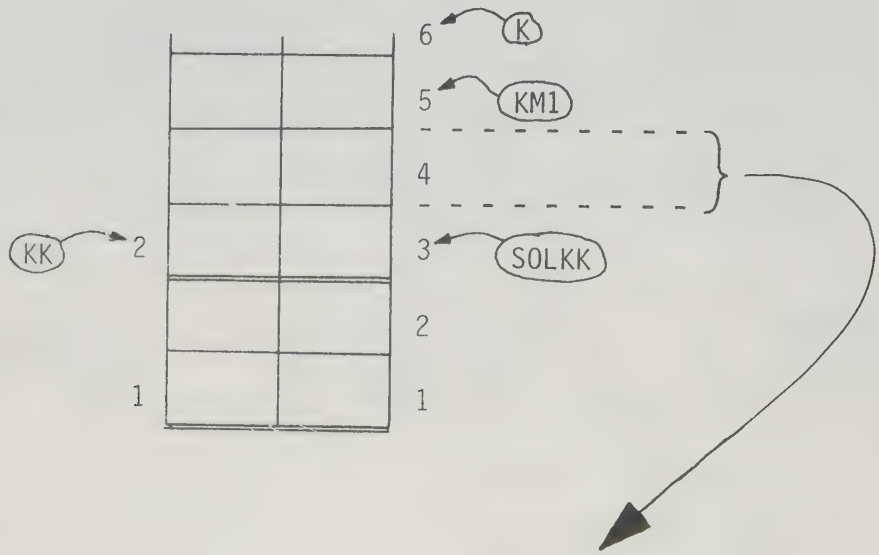
Est-ce la première fois que l'on recule en x ?

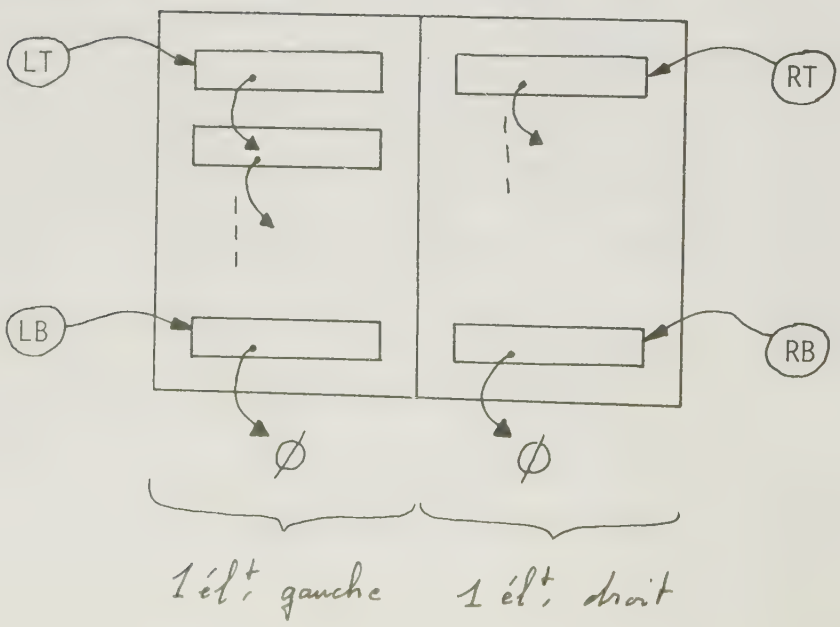
Si oui alors pas de problème , sinon selon que l'arc v est fin ou épais on appelle la procédure "MERGETHIN" ou la procédure "MERGETHICK" .

II/ LA "TWIN UNIT" (TWUN)

1) GENERALITES

La TWUN est constituée d'un empilement de "doubles éléments" , chaque double élément étant formé d'un "élément gauche" et d'un "élément droit".



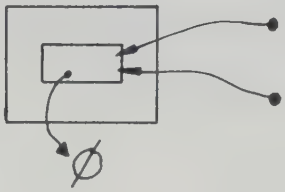


"K" indique toujours le dernier niveau de la TWUN. (avec KM1 = K-1)

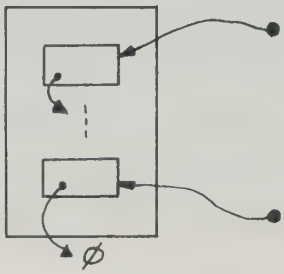
Chaque élément (gauche ou droit) peut être :

- soit vide :

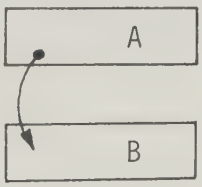
- soit 1 arc :



- soit une liste d'arcs :



Les numéros d'arcs (du coarbre) sont chaînés à l'aide du lien prédécesseur "PRED".



" A a pour prédécesseur B. "
ou
PRED(A) ← B

Le dernier arc (le plus bas) d'un élément a toujours pour prédécesseur le vide.

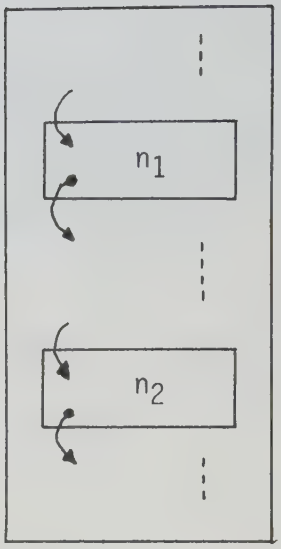
Un élément gauche (resp. droit) n'est accessible que par son haut "Left-Top" (LT) (resp. "Right-Top" ou RT) , et par son bas "Left-Bottom" (LB) (resp. "Right-bottom" ou RB).

2) PREORDRE DANS LA TWIN UNIT

• Relation à l'intérieur d'un élément

Soit n_1 le numéro de l'arc ab.

Soit n_2 le numéro de l'arc cd.



" SI figure respectée
ALORS $b \geqslant d$. "

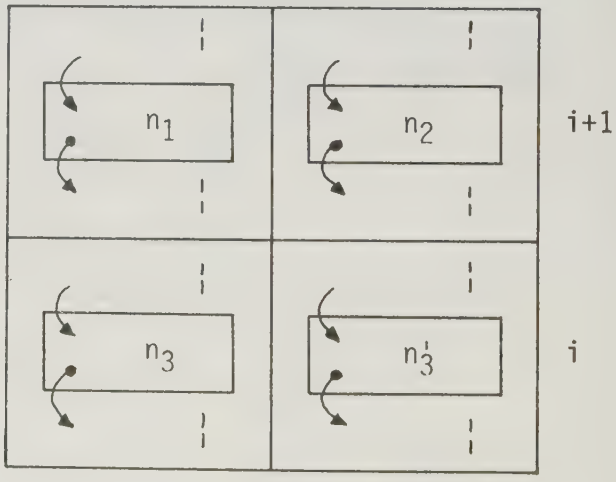
• Relation entre deux doubles éléments consécutifs

n_1 = n°(ab)

n_2 = n°(cd)

n_3 = n°(ef)

(resp. n'_3 = n°(e'f'))



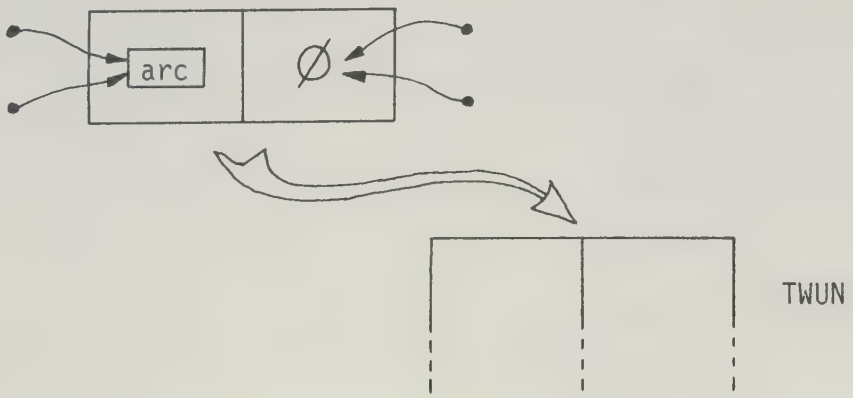
" SI figure respectée
ALORS $f \leqslant b$ ET $f \leqslant d$.
(resp. $f' \leqslant b$ ET $f' \leqslant d$) "

L'étape courante de l'algorithme consiste à ajouter un nouvel arc du coarbre au sommet de la TWUN , et à utiliser éventuellement les 3 règles (que nous allons voir) modifiant la TWUN afin qu'elle vérifie l'ordre partiel décrit ci-dessus.

Avec ces règles , il n'est pas toujours possible de rétablir le préordre , mais dans ce cas et dans ce cas seulement , l'arc que l'on vient d'ajouter rend le graphe déjà construit non-planaire.

• Addition d'un nouvel arc de coarbre

On ajoute en haut de la TWUN un nouveau double élément : son élément gauche contient l'arc en question et son élément droit est vide.



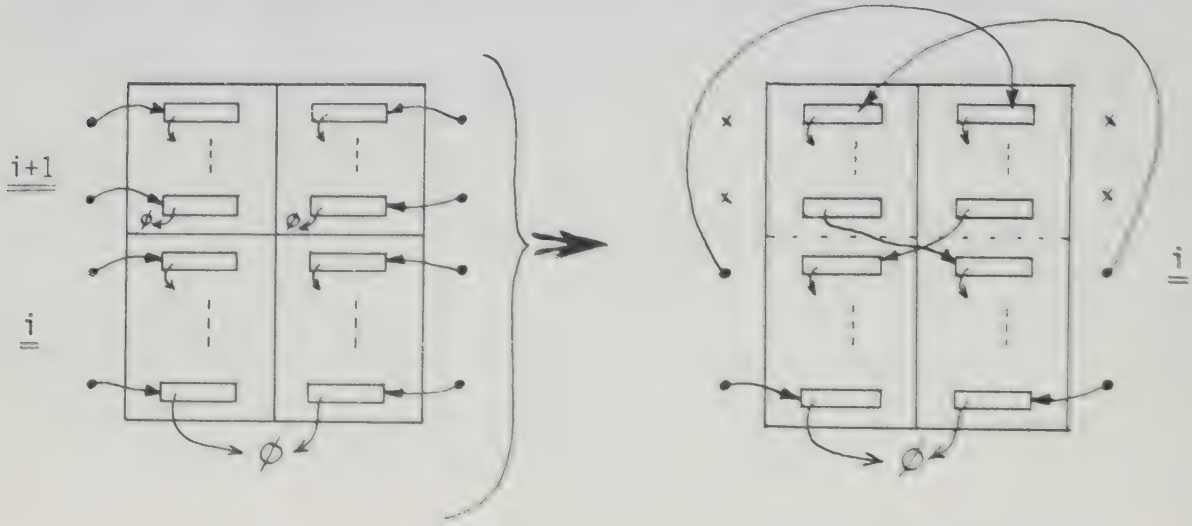
3) DESCRIPTION DES REGLES DE MANIPULATION DE LA TWUN

On peut manipuler la TWUN à l'aide de 3 règles de base :

- La X-fusion
- La H-fusion
- La Bascule (in English : Flipping)

3-1) X-FUSION

Ce mécanisme permet de fusionner les doubles éléments des niveaux "K" et "KM1" , l'élément $gauche_K$ venant sur l'élément $droit_{KM1}$ et lycée de Versailles (c-à-d. l'élément $droit_K$ venant sur l'élément $gauche_{KM1}$). On obtient ainsi un seul double élément.



(Détails des liens lors d'une X-fusion)

(Schéma d'une X-fusion)

---

l'état des 4 éléments concernés lors d'une X-fusion peut être quelconque (soit vide , soit 1 arc , soit une liste d'arcs).

3-2) H-FUSION

Le mécanisme , analogue à celui de la X-fusion , permet de fusionner les 2 couples éléments de niveau "X" et "XM1" , l'élément gauche$_X$ venant alors sur l'élément gauche$_{XM1}$ et ... l'élément droit$_X$ venant sur l'élément droit$_{XM1}$.

(Schéma de la H-fusion)

L'état de chacun des 4 éléments concernés lors d'une H-fusion peut être quelconque.

Remarque: Ces deux mécanismes élémentaires de X-fusion et de H-fusion sont intégralement réalisés par les deux procédures "XFUSION" et "HFUSION" , aussi confondrons nous volontairement les noms des mécanismes et ceux des procédures.

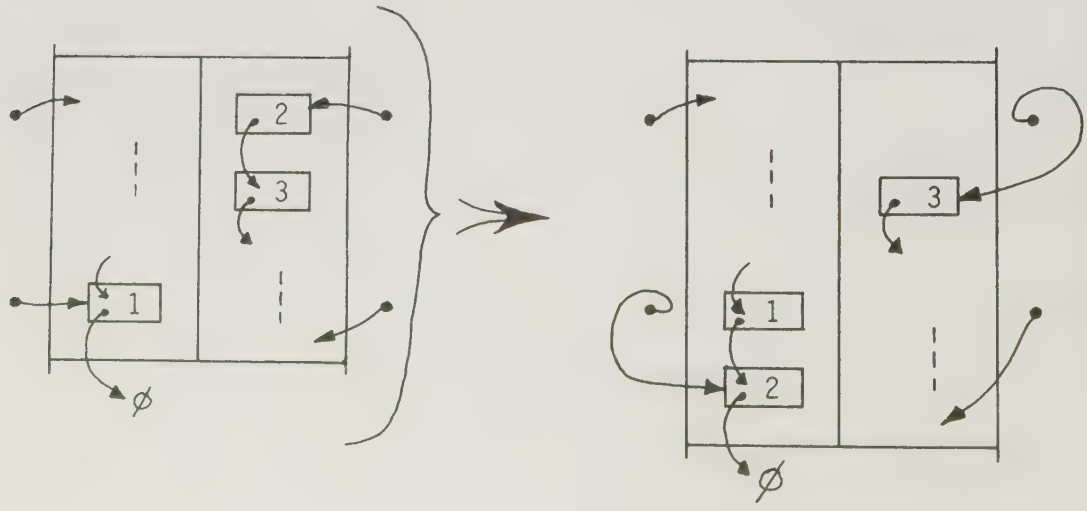
3-3) BASCULE

Ce mécanisme se déroule uniquement à l'intérieur d'un double élément. Nous distinguerons la bascule "Droite-Gauche" et son symétrique la bascule ... "Gauche-Droite" bravo!

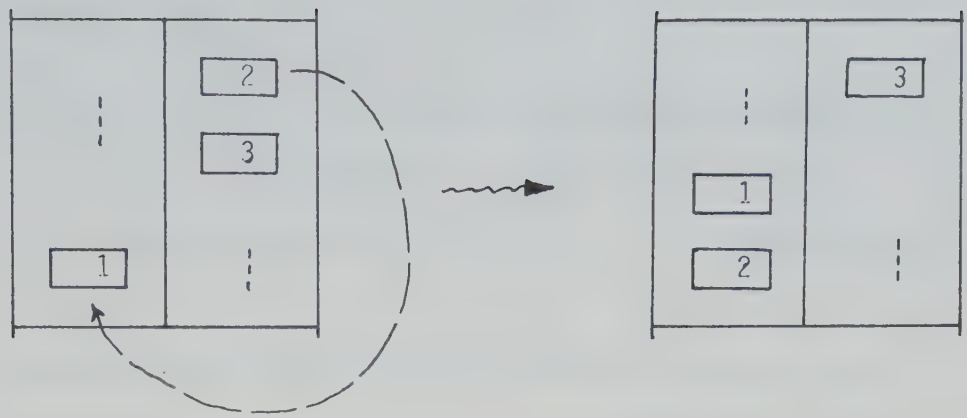
Soit "LEV" (for level) le numéro du double élément concerné.

a) Bascule "Droite-Gauche"

Cette opération élémentaire va consister à ôter l'arc situé en RT(LEV) , et à le placer sous l'arc situé en LB(LEV).



(Détails des liens lors d'une bascule Droite-Gauche)



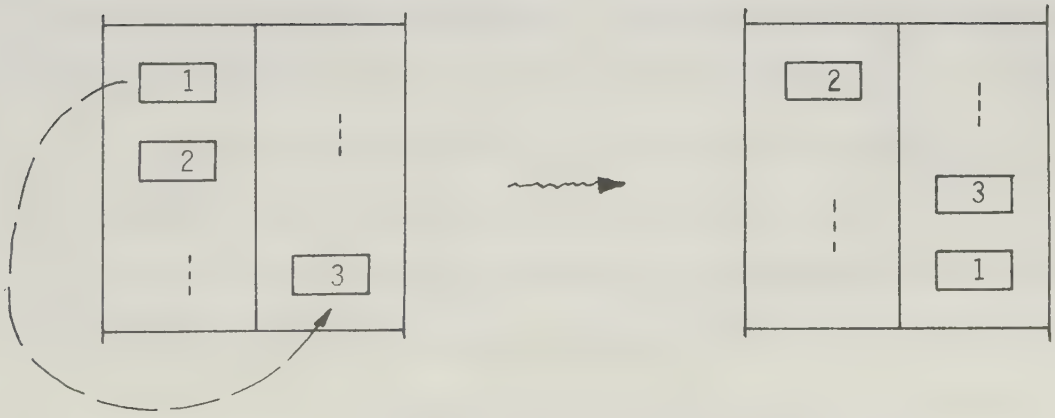
(Schéma d'une bascule Droite-Gauche)

On dit également "bascule de l'arc 2".

---

b) Bascule "Gauche-Droite"

On définit de façon tout à fait analogue cette seconde bascule.



(Schéma d'une bascule Gauche-Droite)

On dit également "bascule de l'arc 1".

Ces deux mécanismes élémentaires sont utilisés dans les procédures "BASCULE1" , "BASCULE2" et "BASCULE3".

III/ DESCRIPTION DE "FLIPPING1"

1) INTRODUCTION

Cette procédure (≡ Bascule1) est appelée uniquement par le sous-programme "FUFLICHOI".

Elle utilise les deux mécanismes élémentaires de bascule (G-D et D-G) et opère toujours dans le double élément de niveau KM1.

2) PARENTHESE

Nous avons vu précédemment que la procédure "FUFLICHOI" choisissait les opérations à faire dans la TWUN pour y rétablir le préordre.

Lorsque la X-fusion ou la H-fusion sont choisis , il n'y a aucun obstacle à leur réalisation : il en va tout autrement pour la procédure "FLIPPING1". Lorsque celle-ci est choisie , ou plus exactement suggérée , le problème n'est pas résolu pour autant : le choix par FUFLICHOI de la procédure Bascule1 ne présume en rien de la 'possibilité' de réaliser la ou les bascules élémentaires à venir.

Il existe en effet une sorte de 'boite noire' , située avant les bascules élémentaires , qui indique après un certain calcul s'il est possible de basculer tel arc choisi : cette 'boite noire' valide (ou invalide) la proposition de bascule.

THEOREME: Si le résultat fournit par cette 'boite noire' est l'invalidation de la bascule proposée , alors le dernier arc du coarbre entré dans la TWUN vient de rendre le graphe déjà construit NON-PLANAIRE.

Preuve: Cf. "A new linear algorithm for planarity testing" ,
by H. de FRAYSSEIX , M. HABIB , J.P. RICHARD , P. ROSENSTIEHL .
(en préparation)

3) ALGORITHME

- Calcul de la variable COVE (servant dans les boites noires)

- SI LV(LT(KM1)) ⩾ LV(RT(KM1))

```
ALORS SI "Boite noire D-G"
        ALORS ┌ TANT QUE (Y < LV(RT(KM1)) FAIRE ;
              │   ⎧
              │   ⎩ Bascule élémentaire Droite-Gauche ;
              │ SI SIDE=G ALORS "XFUSION"
              │           SINON "HFUSION"
              └ RETURN
        SINON SI LV(LT(KM1)) > LV(RT(KM1)) ALORS "TILT" ;
```

- SI "Boite noire G-D"

```
ALORS ┌ TANT QUE (Y < LV(LT(KM1)) FAIRE ;
      │   ⎧
      │   ⎩ Bascule élémentaire Gauche-Droite ;
      │ SI SIDE=G ALORS "HFUSION"
      │           SINON "XFUSION"
      └
SINON "TILT" ;
```

4) REMARQUES

- "TILT" est une étiquette logique correspondant à une sortie brutale de "FLIPPING1" , on 'atterrit' dans le programme principal RIC au niveau : GRAPHE NON PLANAIRE . (perdu ...)

- Chaque bascule élémentaire d'un arc α est comptée sur le Flipping Index : $FI(\alpha) = 1 - FI(\alpha)$, le flipping index étant nul à l'origine.

- Il existe deux boites noires : l'une correspondant à la bascule élémentaire Gauche-Droite , et l'autre correspondant comme il se doit à la bascule élémentaire Droite-Gauche.

  Chacune de ces boites peut se subdiviser en deux parties :

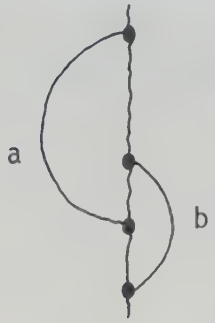
  - une première partie testant l' "admissibilité" de l'arc. Ce test vérifie simplement que l'arc que l'on bascule et donc que l'on introduit sous le bas d'élément ne perturbe pas le préordre.

  - une seconde partie testant la "rigidité" de l'arc.

Hypothèse: "La bascule de l'arc a est demandée (les raisons de cette demande ne sont pas figurées).

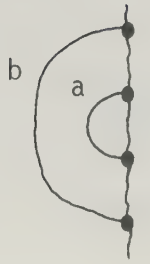
exemple d'arc non-admissible

La bascule de l'arc a est rendue impossible par la présence de l'arc b .



exemple d'arc rigide

La bascule de l'arc a est rendue impossible par la présence de l'arc b .



Cf. "A new linear algorithm for planarity testing" ,
by H. de FRAYSSEIX , M. HABIB , J.P. RICHARD , P. ROSENSTIEHL .
(en préparation)

IV/ DESCRIPTION DE "FUFLICHOI"

1) INTRODUCTION

Procédure FUFLICHOI ? ... FUsion , FLIpping CHOIce procédure.

Cette procédure est appelée uniquement à chaque fois que l'on veut vérifier , et donc rétablir éventuellement , le préordre dans la TWUN entre le dernier double élément (K) et l'avant dernier (KM1).

Elle est appelée depuis "PLANAR" à chaque arrivée d'arc (de coarbre) dans la TWUN , ainsi qu'à la fin de chacun des deux merges.

NARCV: n° de l'arc se trouvant en LB(K) à l'instant de l'appel.

Y: n° du sommet extrémité de cet arc.

2) ALGORITHME

```
TANT QUE (KM1 ⩾ SOLKK) FAIRE
  SI  Y < LV(LT(KM1))
    ALORS  SI  Y < LV(RT(KM1))
             ALORS  "FLIPPING1"
             SINON  SI  SIDE=G  ALORS  "XFUSION"
                                SINON  "HFUSION"
    SINON  SI  Y < LV(RT(KM1))
             ALORS  SI  SIDE=G  ALORS  "HFUSION"
                                SINON  "XFUSION"
             SINON  RETURN
  SI  PRED(NARCV) ≠ ∅  ALORS  RETURN
```

V/ INTERPRETATION GEOMETRIQUE

1) GENERALITES

Dans tous les dessins de graphe suivants , les arcs de l'arbre seront représentés par des segments de droite , et les arcs du coarbre par des segments de courbe.

L'orientation des arcs ne sera pas figurée.

La numérotation des sommets sera celle obtenue lors de la première descente dans le graphe (cf. "TREMAUX").

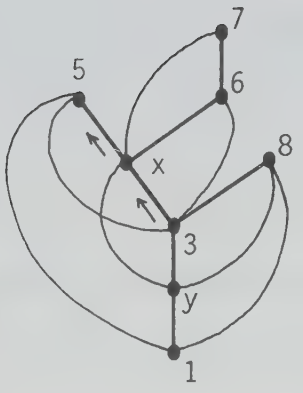
Soit un arc du coarbre (x,y).

Soit le chemin (élémentaire) unique , n'empruntant que des arcs de l'arbre , joignant la racine de l'arbre (≡ sommet n°1) au sommet "x". Remarquons que ce chemin passe obligatoirement par le sommet "y".

Soit l'arc de l'arbre sortant de "x" et étant défini par la procédure "SORTING" comme le premier (des arcs de l'arbre) à emprunter. Cet arc est signalé par une petite flèche sur les dessins.

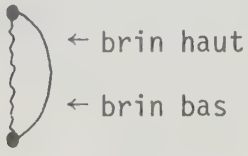
Si nous représentons maintenant ce nouveau chemin d'arcs d'arbre verticalement , la racine en bas , nous définissons ainsi la "gauche" et la "droite" (qui l'eût cru !) : la position de chacun des deux brins de l'arc de coarbre (x,y) sera définie par rapport à ce chemin.

EXEMPLE:



Un trait tremblé représentera un chemin d'arc de l'arbre.

Nous rappelons que le brin correspond à la demi-arête.



Chacun de ces deux brins pouvant être à gauche ou à droite , on obtient ainsi quatre configurations possibles pour un arc de co-arbre :

1

2

3

4

---

2) INTERPRETATION DES MECANISMES ELEMENTAIRES

Nous prendrons un arc (du coarbre) de la TWUN subissant tour à tour une X-fusion , une H-fusion et une bascule.

2-1) X-FUSION

La X-fusion inverse chacun des deux brins de l'arc. On définit également ainsi la position symétrique d'un arc.

Les numéros indiquent les configurations des arcs. (cf. dessin)

1 ⟶ 2	l'arc est dans l'état 1 ; il passe , lors d'une X-fusion , dans l'état 2 .
2 ⟶ 1	
3 ⟶ 4	
4 ⟶ 3	

2-2) H-FUSION

La H-fusion n'engendre aucune modification géométrique des arcs , mais une "solidarisation" (ainsi que la X-fusion d'ailleurs) dont nous reparlerons au chapitre IV .

2-3) BASCULE

Chacune des deux bascules élémentaires conserve la position du brin haut de l'arc , et inverse celle du brin bas.

Bascule "Droite-Gauche" : $\underline{2} \rightarrowtail \underline{4}$

$\underline{3} \rightarrowtail \underline{1}$

Bascule "Gauche-Droite" : $\underline{1} \rightarrowtail \underline{3}$

$\underline{4} \rightarrowtail \underline{2}$

REMARQUE: Tous les arcs (du coarbre) arrivent dans la Twin Unit dans l'état $\underline{1}$.

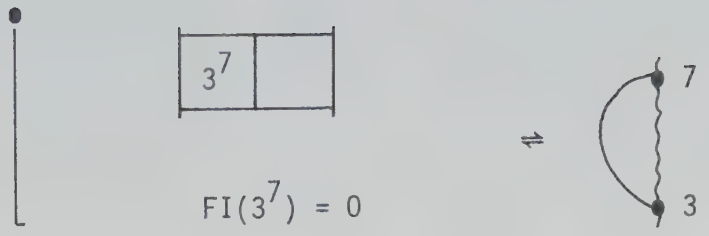
3) INTERPRETATION DE LA "TWUN"

Soit un arc de coarbre (x,y) et α son numéro (celui fourni par la procédure "TREMAUX"). Pour plus de clarté , nous écrirons cet arc dans la TWUN non pas sous la forme α , comme le programme le fait , mais sous la forme y^x .

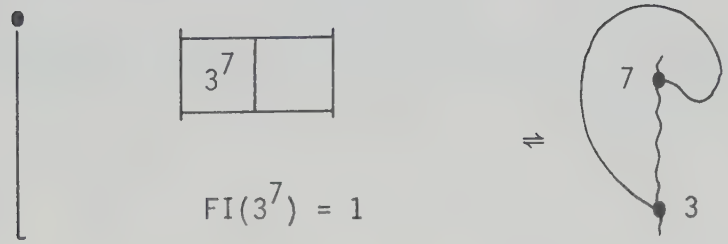
La position G (resp. D) dans la TWUN du numéro de l'arc indique la position géométrique G (resp. G) du brin inférieur (bas) de cet arc.

On récupère la position du brin supérieur (haut) grâce au "Flipping Index" (FI). Si $FI(\alpha) = 0$ alors les deux brins sont du même côté , sinon (c-à-d. $FI(\alpha) = 1$) les deux brins sont de côté opposé.

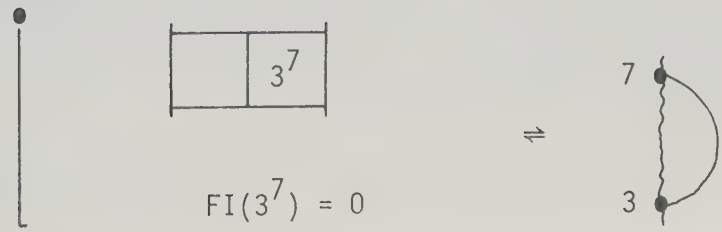
EXEMPLE:



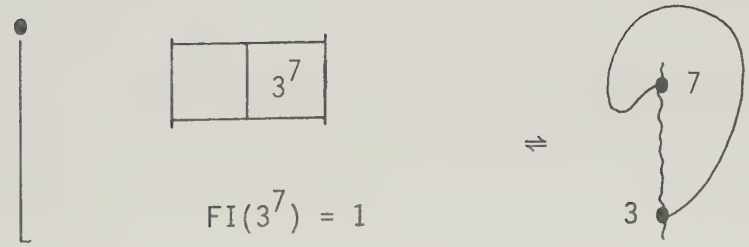
$FI(3^7) = 0$



$FI(3^7) = 1$



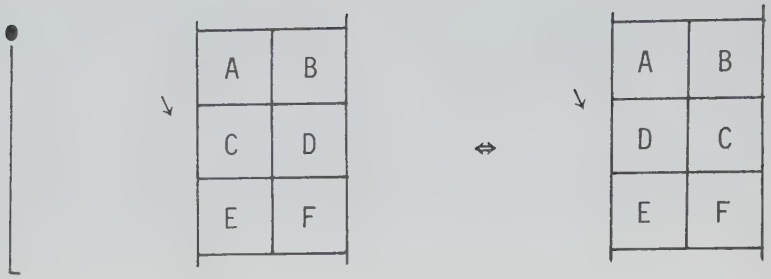
$FI(3^7) = 0$



$FI(3^7) = 1$

Un double élément de la TWUN peut être inversé , à condition de le prendre dans son ensemble , en entier.

Les deux parties de TWUN dessinées sont absolument équivalentes. La position G ou D à l'intérieur d'un double élément est relative et non pas absolue.



Ceci revient à dire que les arcs d'un double élément sont solidaires : si l'on fixe un côté pour l'un d'entre eux , alors le côté de chacun des autres arcs de ce double élément est imposé.

Nous reviendrons au chapitre IV sur cette notion.

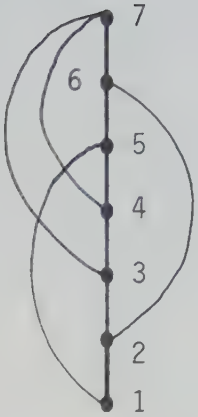
Retenons pour l'instant qu'il existe pour chaque double élément , une liberté de "rotation".

4) EXEMPLE

Nous allons maintenant exécuter l'algorithme " à la main " sur un petit graphe , en traitant parallèlement l'aspect géométrique.

Nous supposerons la première partie de l'algorithme (c-à-d. "SORTING" et "TREMAUX") effectuée , et donc les sommets renumérotés et l'arbre de Trémaux construit.

Soit le graphe de 7 sommets et de 10 arêtes , dont l'arbre trouvé est un chemin hamiltonien , suivant :



On vérifie facilement que , une fois triés par "SORTING" , les arcs du coarbre se présenteront dans "PLANAR" dans l'ordre suivant : (7,3) , (7,4) , (6,2) et (5,1) .

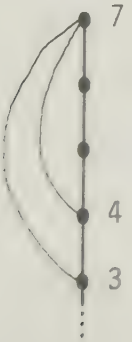
1ère étape: arc (7,3)

3^7	$\emptyset$

1

Le préordre est respecté , on passe à l'étape suivante.

2ième étape: arc (7,4)



4^7	$\emptyset$
3^7	$\emptyset$

2
1

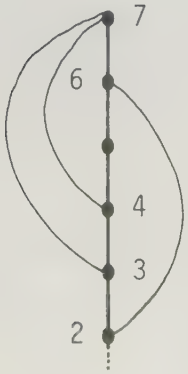
Le préordre est toujours respecté , on passe à l'étape suivante.

3ième étape: arc (6,2)

2^6	$\emptyset$	3
4^7	$\emptyset$	2
3^7	$\emptyset$	1

Le préodre n'est plus respecté entre les niveaux 2 et 3. (comme c'est bizarre et quelle coïncidence ... le graphe n'est plus planaire !)

La procédure "FUFLICHOI" va choisir d'effectuer une X-fusion entre ces deux niveaux ...



4^7	2^6	2
3^7	$\emptyset$	1

Le préordre n'est toujours pas respecté , cette fois entre les niveaux 1 et 2.

La procédure "FUFLICHOI" va opter pour une H-fusion entre ces deux niveaux ...

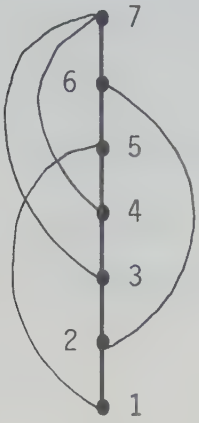
(même dessin)

4^7 3^7	2^6	1

Maintenant le préordre est respecté.

On remarquera que ces trois arcs , étant dans le même double élément , sont devenus solidaires. (ce que nous indiquait le dessin)

4ième étape: arc (5,1)

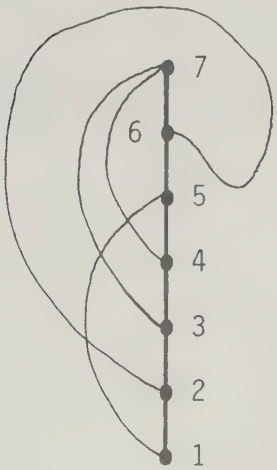


1^5	Ø	2
4^7 3^7	2^6	1

Le préordre , à nouveau , n'est pas respecté. La procédure "FUFLICHOI" va choisir d'effectuer "FLIPPING1" (sur le niveau 1).

La procédure "FLIPPING1" va déclencher le mécanisme élémentaire de bascule Droite-Gauche de l'arc (6,2).

Nous admettrons pour l'instant que la boite noire D-G va valider cette demande d'opération.

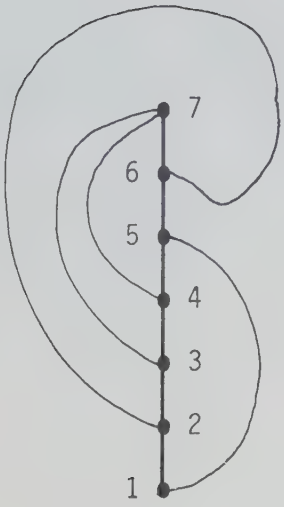


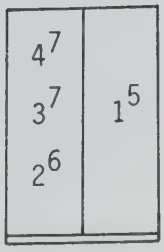
1^5	Ø	2
4^7 3^7 2^6	Ø	1

avec $FI(2^6) = 1$

Nous sommes toujours dans la procédure "FLIPPING1" , qui fait suivre cette bascule élémentaire d'une X-fusion.

Les quatre arcs du coarbre vont ainsi devenir solidaires.





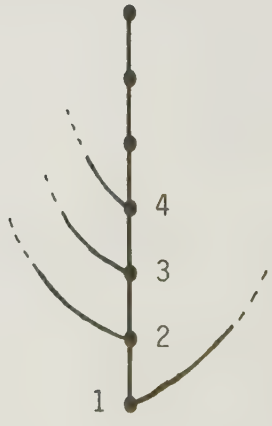
avec $FI(2^6) = 1$

Le préordre est enfin respecté.

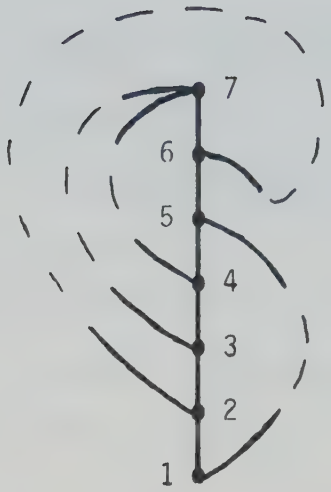
Tous les arcs du coarbre ayant pu être placés dans la Twin Unit , tout en respectant le préordre devant y régner , ce graphe sera (il reste encore à effectuer le parcours de l'arbre qui nous ramènera à la racine) déclaré planaire.

Montrons qu'avec la TWUN , dans l'état ci-dessus , nous pouvons retrouver le dessin :

Positionnons d'abord les brins bas , conformément à la TWUN.



Récupérons maintenant les brins hauts : ils sont tous du même côté que les brins bas excepté pour l'arc (6,2) qui a été basculé.



VI/ DESCRIPTION DES "MERGES"

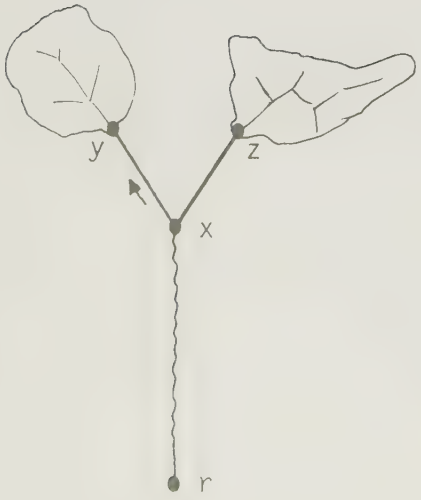
1) INTRODUCTION

Jusqu'à présent , nous avons toujours travaillé sans le dire , sur des graphes dont l'arbre de Trémaux trouvé était un chemin hamiltonien. Nous savons le fonctionnement de l'algorithme dans ce cas , les mécanismes mis en jeu ainsi que la représentation du graphe sous-jacente.

Nous allons maintenant traiter les extensions de l'algorithme dans le cas d'un arbre de Trémaux quelconque.

2) RETOUR SUR L'ALGORITHME

Soient
- r la racine de l'arbre ;
- x un sommet origine d'au moins 2 arcs de l'arbre : (x,y) et (x,z); (r ... x) est un chemin élémentaire n'empruntant que des arcs de l'arbre.



La procédure "SORTING" donnera l'ordre de traitement des 2 arcs (x,y) et (x,z).

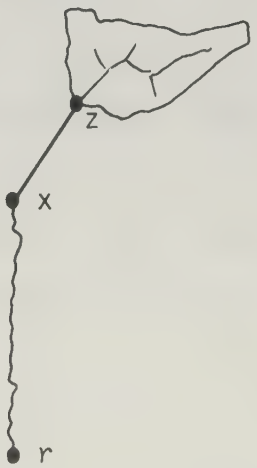
Supposons pour fixer les idées que l'arc (x,y) soit à traiter avant l'arc (x,z).

Dans un premier temps , l'algorithme va traiter ce sous-arbre , et ceci de la manière que nous connaissons.

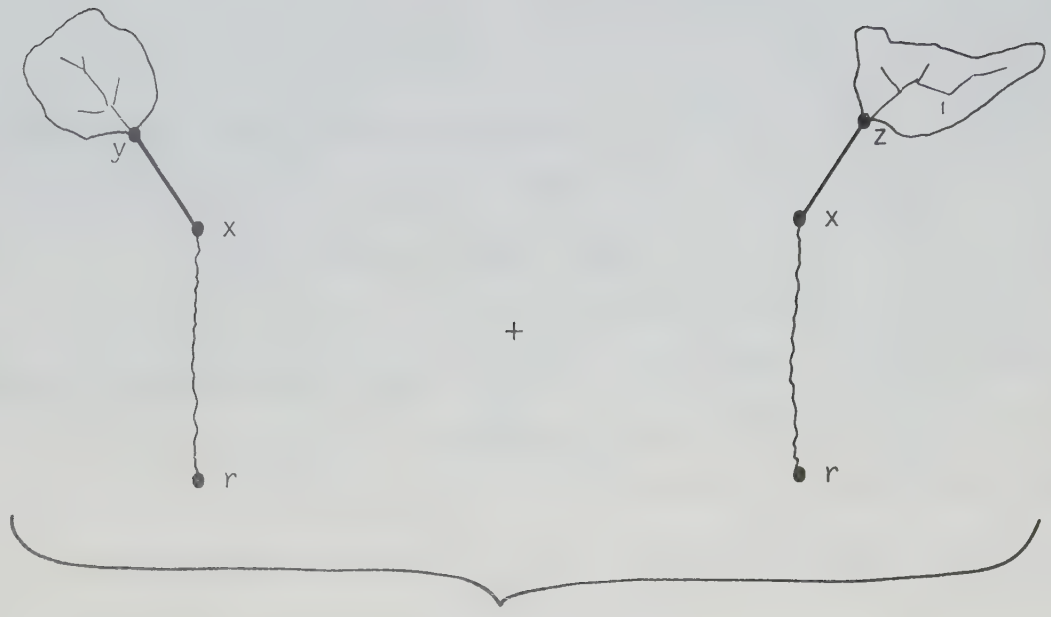
Il va y avoir descente dans le graphe puis retour en x , venant de y.

Nous arrivons ainsi au deuxième temps : passage sur l'arc (x,z). Tout ceci moyennant l'ordre de passage des arcs de l'arbre et du co-arbre , lequel ordre est défini par "SORTING".

Il va y avoir poursuite de la descente dans le graphe , puis retour en x , venant de z.



Ce second retour en x déclenche le troisième temps : il va falloir 'souder' ces deux morceaux.

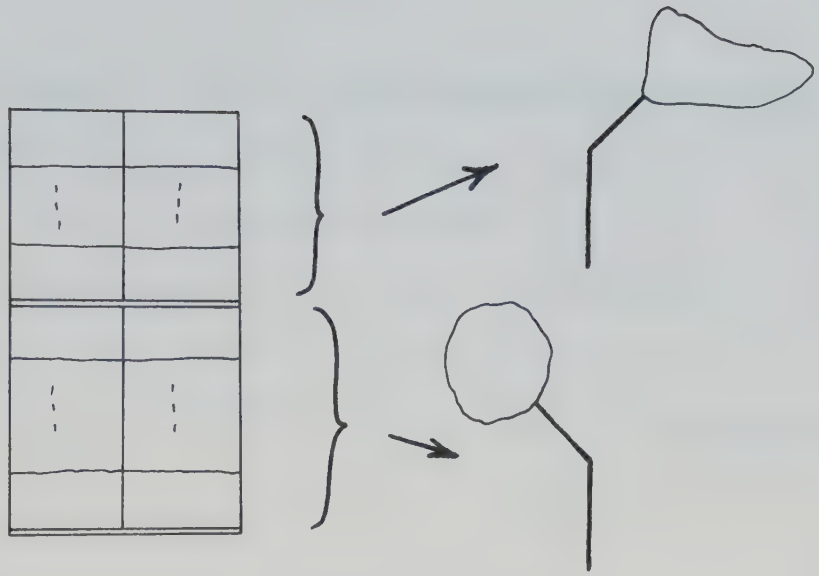


Cette 'fusion' peut engendrer de nouvelles anomalies dans le préordre : c'est ici le point de différence avec le cas où l'arbre est un chemin hamiltonien.

Il faudra rétablir le préordre impérativement avant de poursuivre ; dans le cas d'une impossibilité , le graphe initial serait alors déclaré NON-PLANAIRE.

Lorsque l'on emprunte l'arc (x,y) , on 'ouvre' un plancher de type KK dans la TWUN. (cf. schéma TWUN)

On ignore alors volontairement toute la partie de la TWUN située sous ce plancher.



La "fusion" des deux parties consistera en la suppression de ce plancher KK , son remplacement par un plancher K 'simple' , et donc la prise en compte de toute la partie de la TWUN située au-dessus du précédent plancher KK , partie qui était jusqu'alors ignorée.

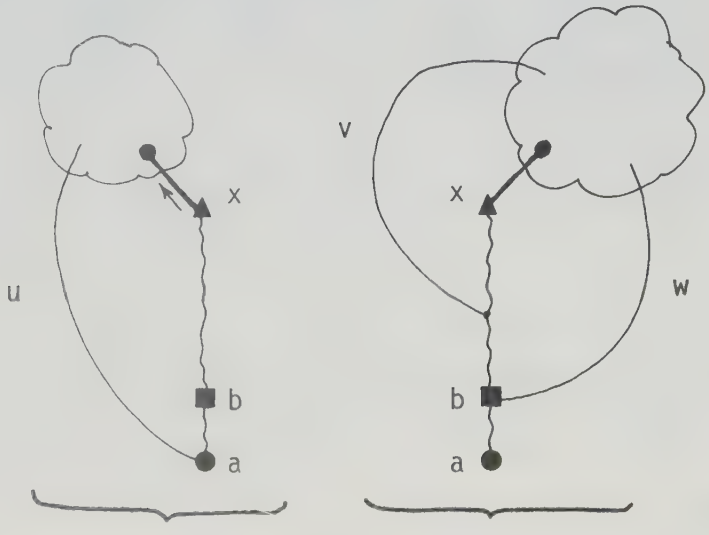
Cette "fusion" peut engendrer des perturbations dans le préordre de la TWUN , et donc son rétablissement.

Cette "fusion" , que nous appellerons dorénavant "merge" , sera réalisée par les procédures "MERGETHIN" et "MERGETHICK" , selon que l'arc (x,y) est mince ou épais.

Les trois mécanismes élémentaires de manipulation de la TWUN que nous connaissons déjà sont les seuls à être utilisés lors d'un merge pour rétablir le préordre dans la TWUN.

PROPRIETE 1: Pour qu'un MERGE aboutisse , il est nécessaire (mais non suffisant) de pouvoir positionner dans la TWUN , tous les arcs de la branche dernièrement parcourue d'un côté et d'un seul.

Démonstration



- Soit "u" l'arc de coarbre de la branche A de plus petite extrémité "a".
- Soit "v" (resp. "w") l'arc de coarbre de la branche B placé à gauche (resp. à droite) de plus petite extrémité.
- Soit "b" le minimum des extrémités de v et w.

Nous savons que $a \leqslant b$ d'après la procédure "SORTING".

Nous avons , entre autre , à placer l'arc "u". L'origine de cet arc est relié au sommet "x" par un chemin élémentaire dans l'arbre.

Nous avons donc un chemin [x...a] : symbolisons le par un arc de coarbre fictif (x,a).

Il est évident que l'arc "v" (resp. "w") va gèner si l'on place l'arc (x,a) à gauche (resp. à droite) , et avec l'arc "v" (resp. "w") tous les arcs de la branche B placés du même côté (et descendant moins bas).

Il faudra donc libérer complètement un côté du 'tronc' avant de faire le merge. (c-à-d. changer au moins les brins bas de côté à l'aide des mécanismes élémentaires que nous connaissons.)

MERGETHIN

- Positionner à gauche tous les arcs du double élément supérieur de niveau K.

- TANT QUE K > SOL(KK) FAIRE
  - Positionner à gauche tous les arcs du double élément de niveau KM1 , et 'souder' le double élément "K" au double élément "KM1".
  - K ← K-1 ; KM1 ← K-1 ;

- Parcourir ce double élément "K" et positiooner les variables "DUS" et "ERA".

- Ne conserver que l'arc du bas (LB) gauche dans ce double élément "K" , et appeler la procédure "FUFLICHOI" avec cet arc.

FIN

MERGETHICK

- Calcul de la variable "COVE". (servant pour la détermination de la 'rigidité' ; cette variable est unique pour un appel à "MERGETHICK".)

- Positionner à gauche tous les arcs du double élément supérieur de niveau K.

- TANT QUE K > SOL(KK) FAIRE
  - Positionner à gauche tous les arcs du double élément de niveau KM1 , et 'souder' le double élément "K" au double élément "KM1".
  - K ← K-1 ; KM1 ← K-1 ;

- Appeler la procédure "FUFLICHOI" avec l'arc situé en LB(K).

FIN

Chacun des deux merges peut se résumer ainsi:

1-Dégagement du 'tronc'

Parcours de la Twin Unit , du haut (niveau "K") au dernier plancher KK , en positionnant à gauche les arcs , et en fusionnant les doubles éléments.

2-'Soudure' des deux branches

Appeler la procédure "FUFLICHOI" avec l'arc situé en LB(K).

PROPRIETE 2: Dans l'algorithme des MERGE , on peut remplacer la partie 'dégagement du tronc' par l'adjonction dans la TWUN , au même instant , de l'arc de coarbre fictif (x,a). (c-à-d. appeler la procédure "FUFLICHOI" avec l'arc (x,a).)

Démonstration: L'arc (x,a) descendant plus bas que tous les arcs de la branche B "forcera" ceux-ci d'un côté , par cascade successives dans la TWUN.

REMARQUE 1: Dans le programme nous avons arbitrairement choisi de mettre tous les arcs de coarbre de la branche B du côté gauche.

REMARQUE 2: Dans le cas d'un "MERGETHIN" , tous les arcs de coarbre de la branche B aboutissant au même sommet (du tronc) , leur positionnement unilatéral est toujours réalisable.

REMARQUE 3: Nous avons choisi de rédiger la partie 'dégagement du tronc' , dans les deux procédures merges , au lieu d'utiliser les possibilités de la propriété 2 , dans un soucis d'efficacité: le travail est fait à moindre frais , en utilisant les informations que nous avons , informations qui ne seraient pas exploitées si nous utilisions la 'machinerie' classique.

Ceci nous a entre autres conduit à rédiger deux nouvelles procédures de bascule: "FLIPPING2" (appelée par la procédure "MERGETHIN") et "FLIPPING3" (appelée par la procédure "MERGETHICK") , plus spécialisée que la procédure générale "FLIPPING1".

CHAPITRE II

LISTING DU PROGRAMME ET EXEMPLES

I/ INTRODUCTION

Ce programme présente l'algorithme "Gauche-Droite" complet , à savoir : test de planéarité , et si oui , liste des arcs de l'arbre et du coarbre , avec pour ceux-ci la position de chaque brin.

Nous avons voulu ce programme simple , clair , de même architecture que l'algorithme , distinguant toutes les différentes étapes et facilitant la compréhension de l'algorithme , et enfin si possible performant.

Tout ceci étant quelque peu contradictoire , partant de cet ambitieux projet nous avons abouti à ce programme tout à fait bâtard qui n'est malheureusement pas le programme le plus simple , ni le programme le plus court , ni bien sûr le programme le plus efficace.

Il faut savoir que la place mémoire a été plus volontiers sacrifiée à la clarté que le temps de calcul.

Enfin , dans ce même soucis de clarté et de lisibilité , le langage choisi a été le PL1 ; toutes les procédures sont externes.

Le graphe d'entrée est supposé SIMPLE et CONNEXE , ceci n'étant pas vérifié par le programme.

Le graphe est donné sous la forme suivante :

1) "N" et "M"

2) Enumération des arêtes

PROGRAM FOR PLANARITY TESTING AND EMBEDDING OF A CONNECTED SIMPLE GRAPH

```
00001 /*****************************************************************************/
00002 /* PROGRAM FOR PLANARITY TESTING AND EMBEDDING OF A CONNECTED SIMPLE GRAPH.  */
00003 /* 'LEFT-RIGHT' ALGORITHM , IN LINEAR TIME AND SPACE , DEVISED BY            */
00004 /*   H. DE FRAYSSEIX  AND  P. ROSENSTIEHL (PARIS).                           */
00005 /* (VERSION C1 WRITTEN BY J-P. RICHARD AND D. TALLOT)(JAN-82,JUSSIEU,PARIS)  */
00006 /*****************************************************************************/
00007
00008 RIC: PROCEDURE OPTIONS(MAIN) ;
00009
00010 DCL  GRAF  FILE STREAM INPUT ;
00011 DCL  (TREMAUX,SORTING,PLANAR,EMBED)  ENTRY ;
00012 DCL  (N,M,NM1,NP1,NP2M,MP1)  BIN FIXED(15,0) ;
00013 DCL  TILT  LABEL  EXTERNAL ; TILT=FIN2 ;
00014
00015 OPEN FILE(GRAF) ; GET FILE(GRAF) EDIT(N,M)(SKIP,2 F(6)) ;
00016 NM1=N-1 ; NP1=N+1 ; NP2M=N+(2*M) ; MP1=M+1 ;
00017 PUT SKIP(4) EDIT('NUMBER OF VERTICES:',N,'NUMBER OF EDGES:',M)
00018                 (A(19),F(5),X(12),A(16),F(5)) ; PUT SKIP(4) ;
00019 IF (N<=4)!(M=NM1) THEN GOTO FIN1 ;
00020                   ELSE IF M>(3*N)-6 THEN GOTO FIN2 ;
00021
00022 BEGIN; DCL  (UV,LV)(1:MP1)  BIN FIXED(15,0) ,
00023             ONU(1:N)        BIN FIXED(15,0) ,
00024             TYPE(1:NM1)     BIN FIXED(15,0) INIT((NM1)1) ,
00025             (X,Y,I,J)       BIN FIXED(15,0) ,
00026             SL(1:N+M)       BIN FIXED(15,0) INIT((N+M)0) ,
00027             NARC(NP1:N+M)   BIN FIXED(15,0) ;
00028 BEGIN; DCL  LOW(1:NM1)  BIN FIXED(15,0) ;
00029 BEGIN; DCL  SOR(1:NP2M)           BIN FIXED(15,0) INIT((NP2M)0) ,
00030             (SORM1,OR)(NP1:NP2M)  BIN FIXED(15,0) ,
00031             EX(N:NP2M+1)          BIN FIXED(15,0) ;
00032
00033   /** READING OF EDGES  &  GRAPH CONSTRUCTION **/
00034 BEGIN; DCL ID(1:N) BIN FIXED(15,0) ;
00035 DO I=1 TO N BY 1; ID(I)=I ; END;
00036 EX(N),EX(NP2M+1)=0 ;
00037 DO I=NP1 TO NP2M BY 2;
00038   GET FILE(GRAF) EDIT(X,Y)(SKIP,2 F(6)) ;
00039   SOR(ID(X))=I ; SORM1(I)=ID(X) ; ID(X)=I ; EX(I)=Y ; OR(I)=X ; J=I+1 ;
00040   SOR(ID(Y))=J ; SORM1(J)=ID(Y) ; ID(Y)=J ; EX(J)=X ; OR(J)=Y ;
00041 END; END; CLOSE FILE(GRAF) ;
00042
00043 CALL TREMAUX (SOR,SORM1,EX,OR,N,NM1,UV,LV,LOW,TYPE,ONU) ;
00044 UV(NP1)=NP1 ; LV(NP1)=0 ;
00045 END;
00046
00047 CALL SORTING (N,M,NM1,UV,LV,TYPE,LOW,SL,NARC) ;
00048 END;
00049
```

```
00050 BEGIN; DCL  PRED(N:M)       BIN FIXED(15,0) INIT((M-NM1)MP1) ,
00051             FI(N:MP1)       BIN FIXED(15,0) INIT((MP1-NM1)0) ,
00052             (DUS,ERA)(N:M)  BIN FIXED(15,0) INIT((M-NM1)0) ;
00053
00054 CALL PLANAR (N,M,SL,NARC,UU,LU,TYPE,FI,PRED,DUS,ERA) ;
00055 BEGIN; DCL  (US,LS,LIB)(N:M)  BIN FIXED(15,0) ;
00056
00057 CALL EMBED (N,M,MP1,PRED,FI,DUS,ERA,US,LS,LIB) ;
00058
00059 PUT SKIP LIST('**** TREE ARCS ****') ;
00060 PUT SKIP(3) EDIT('( X , Y )')(X(5),A(9)) ; PUT SKIP ;
00061 DO I=1 TO NM1 BY 1;
00062   X=ONU(LU(I)) ; Y=ONU(UU(I)) ;
00063   PUT SKIP EDIT('(',X,',',Y,')')(X(5),2(A(1),F(3)),A(1)) ;
00064 END;
00065 PUT SKIP(5) LIST('**** COTREE ARCS ****') ;
00066 PUT SKIP(3) EDIT('( X , Y )','US','LS','LIB')
00067                (X(5),A(9),X(11),A(2),X(4),A(2),X(8),A(3)) ; PUT SKIP ;
00068 DO I=N TO M BY 1;
00069   X=ONU(UU(I)) ; Y=ONU(LU(I)) ;
00070   PUT SKIP EDIT('(',X,',',Y,')',US(I),LS(I),LIB(I))
00071                (X(5),2(A(1),F(3)),A(1),X(12),F(1),X(5),F(1),X(9),F(2)) ;
00072 END; END; END; END;
00073
00074 FIN1: PUT SKIP(5) LIST('** BRAVO !    THIS GRAPH IS PLANAR **') ; GOTO FIN3 ;
00075 FIN2: PUT SKIP(5) LIST('** LOST...    THIS GRAPH IS NOT PLANAR **') ;
00076 FIN3: PUT PAGE ; END RIC;
```

```
00077   /*********************************/
00078   /* CONSTRUCTION OF A TREMAUX TREE */
00079   /*********************************/
00080
00081 TREMAUX: PROCEDURE (SOR,SORM1,EX,OR,N,NM1,UU,LU,LOW,TYPE,ONU) ;
00082
00083 DCL  ((SOR,SORM1,EX,OR,UU,LU,LOW,TYPE,ONU)(*),N,NM1)  BIN FIXED(15,0) ;
00084 DCL  (STACK,NUM)(1:N)                                 BIN FIXED(15,0) ,
00085      VISIT(1:N)                                      BIN FIXED(15,0) INIT((N)0) ,
00086      SLOW(1:NM1)                                     BIN FIXED(15,0) ,
00087      (V,W,X,Y,P,NUMY,ARC,ARCP,TPT,COTPT)             BIN FIXED(15,0) ;
00088
00089 TPT,P=0 ; COTPT=NM1 ; NUM(1),ONU(1),VISIT(1)=1 ; V=SOR(1) ;
00090
00091 A: DO WHILE (V^=0);
00092      X=OR(V) ; Y=EX(V) ;
00093      IF X^=EX(V-1) THEN W=V+1 ;
00094                    ELSE IF Y^=OR(V-1) THEN W=V+1 ;
00095                                       ELSE W=V-1 ;
00096      IF SOR(W)=0 THEN SOR(SORM1(W))=0 ;
00097                  ELSE DO; SOR(SORM1(W))=SOR(W) ; SORM1(SOR(W))=SORM1(W) ; END;
00098      IF VISIT(Y)=0
00099        THEN DO;/* (XY): Tree Arc */
00100               VISIT(Y)=1 ; P=P+1 ; STACK(P)=V ;
00101               NUM(Y),NUMY=TPT+2 ; ONU(NUMY)=Y ; ARC=NUMY-1 ;
00102               LOW(ARC),SLOW(ARC)=NUMY ;
00103               TPT=TPT+1 ; LU(TPT)=NUM(X) ; UU(TPT)=NUMY ; V=SOR(Y) ;
00104             END;
00105        ELSE DO;/* (XY): Cotree Arc */
00106               NUMY=NUM(Y) ; ARC=NUM(X)-1 ;
00107               IF NUMY<LOW(ARC)
00108                 THEN DO; SLOW(ARC)=LOW(ARC) ; LOW(ARC)=NUMY ; END;
00109                 ELSE IF LOW(ARC)<NUMY & NUMY<SLOW(ARC) THEN SLOW(ARC)=NUMY ;
00110               COTPT=COTPT+1 ; UU(COTPT)=NUM(X) ; LU(COTPT)=NUMY ; V=SOR(V) ;
00111             END;
00112 END A;
00113
00114 IF P>1 THEN DO;/* backing on the tree */
00115               V=STACK(P) ; P=P-1 ; ARC=NUM(OR(V))-1 ; ARCP=NUM(EX(V))-1 ;
00116               IF LOW(ARCP)<LOW(ARC)
00117                 THEN DO; IF SLOW(ARCP)<LOW(ARC) THEN SLOW(ARC)=SLOW(ARCP) ;
00118                                                 ELSE SLOW(ARC)=LOW(ARC) ;
00119                          LOW(ARC)=LOW(ARCP) ;
00120                      END;
00121                 ELSE IF LOW(ARCP)=LOW(ARC)
00122                        THEN DO; IF SLOW(ARCP)<SLOW(ARC)
00123                                   THEN SLOW(ARC)=SLOW(ARCP) ;
00124                                 END;
00125                        ELSE IF LOW(ARCP)<SLOW(ARC) THEN SLOW(ARC)=LOW(ARCP) ;
00126               V=SOR(V) ; GOTO A ;
00127             END;
00128        ELSE IF P=1 THEN DO; V=SOR(STACK(1)) ; P=0 ; GOTO A ; END;
00129
00130 DO ARC=1 TO NM1 BY 1; IF SLOW(ARC)>ARC THEN TYPE(ARC)=0 ; END;
00131
00132 END TREMAUX;
```

```
00133   /*********************************/
00134   /* SORTING ALL ARCS OF THE GRAPH */
00135   /*********************************/
00136
00137 SORTING: PROCEDURE (N,M,NM1,UU,LU,TYPE,LOU,SL,NARC) ;
00138
00139 DCL  (N,M,NM1,(UU,LU,TYPE,LOU,SL,NARC)(*))  BIN FIXED(15,0) ;
00140 DCL  (I,X,X1,X2,X3,VAR,ARC,U)               BIN FIXED(15,0) ,
00141      (TNL,TKL)(1:2*N-1)                     BIN FIXED(15,0) INIT((2*N-1)0) ,
00142      CTL(1:M+1)                             BIN FIXED(15,0) INIT((M+1)0) ,
00143      (TNLINU,TKLINU,TNN,TKN)(N+1:2*N-1)     BIN FIXED(15,0) ,
00144      CTN(N+1:M+1)                           BIN FIXED(15,0) ,
00145      (SLI,CTLI,TNLI,TKLI)(1:N)              BIN FIXED(15,0) ;
00146
00147 X,X1,X2,X3=N ; DO I=1 TO N BY 1; SLI(I),CTLI(I),TNLI(I),TKLI(I)=I ; END;
00148
00149
00150 /** SORTING -THIN & THICK TREE ARCS ACCORDING TO THEIR "LOW" **/
00151 /**         -COTREE ARCS ACCORDING TO THEIR "LU"             **/
00152
00153 DO I=1 TO NM1 BY 1; /* tree arc */
00154   VAR=LOU(I) ;
00155   IF TYPE(I)=0 THEN DO;/* thin arc */
00156                        X2=X2+1 ; TNL(TNLI(VAR))=X2 ; TNLINU(X2)=TNLI(VAR) ;
00157                        TNLI(VAR)=X2 ; TNN(X2)=I ;
00158                     END;
00159                ELSE DO;/* thick arc */
00160                        X3=X3+1 ; TKL(TKLI(VAR))=X3 ; TKLINU(X3)=TKLI(VAR) ;
00161                        TKLI(VAR)=X3 ; TKN(X3)=I ;
00162                     END;
00163 END;
00164
00165 DO I=N TO M BY 1; /* cotree arc */
00166   VAR=LU(I) ;
00167   X1=X1+1 ; CTL(CTLI(VAR)),CTLI(VAR)=X1 ; CTN(X1)=I ;
00168 END;
00169
```

```
00170
00171 /** FILLING THE SORTING LIST **/
00172
00173 DO I=1 TO N BY 1;
00174   VAR=TNL(I) ;/* Thin Arc */
00175   DO WHILE (VAR^=0);
00176     ARC=TNN(VAR) ; U=LU(ARC) ;
00177     IF SL(U)=0 THEN DO; X=X+1 ; SL(U),SLI(U)=X ; NARC(X)=ARC ;
00178                         IF TNL(VAR)=0 THEN TNL(TNLINU(VAR))=0 ;
00179                                       ELSE DO; TNL(TNLINU(VAR))=TNL(VAR) ;
00180                                                TNLINU(TNL(VAR))=TNLINU(VAR) ;
00181                                            END;
00182                       END;
00183     VAR=TNL(VAR) ;
00184   END;
00185
00186   VAR=TKL(I) ;/* Thick Arc */
00187   DO WHILE (VAR^=0);
00188     ARC=TKN(VAR) ; U=LU(ARC) ;
00189     IF SL(U)=0 THEN DO; X=X+1 ; SL(U),SLI(U)=X ; NARC(X)=ARC ;
00190                         IF TKL(VAR)=0 THEN TKL(TKLINU(VAR))=0 ;
00191                                       ELSE DO; TKL(TKLINU(VAR))=TKL(VAR) ;
00192                                                TKLINU(TKL(VAR))=TKLINU(VAR) ;
00193                                            END;
00194                       END;
00195     VAR=TKL(VAR) ;
00196   END;
00197 END;
00198
00199 DO I=1 TO N BY 1; /* OTHERS ARCS */
00200   VAR=CTL(I) ;
00201   DO WHILE (VAR^=0);
00202     ARC=CTN(VAR) ; U=UU(ARC) ;
00203     X=X+1 ; SL(SLI(U)),SLI(U)=X ; NARC(X)=ARC ; VAR=CTL(VAR) ;
00204   END;
00205
00206   VAR=TNL(I) ;
00207   DO WHILE (VAR^=0);
00208     ARC=TNN(VAR) ; U=LU(ARC) ;
00209     X=X+1 ; SL(SLI(U)),SLI(U)=X ; NARC(X)=ARC ; VAR=TNL(VAR) ;
00210   END;
00211
00212   VAR=TKL(I) ;
00213   DO WHILE (VAR^=0);
00214     ARC=TKN(VAR) ; U=LU(ARC) ;
00215     X=X+1 ; SL(SLI(U)),SLI(U)=X ; NARC(X)=ARC ; VAR=TKL(VAR) ;
00216   END;
00217 END;
00218
00219
00220 END SORTING;
```

```
00221   /*********************************/
00222   /* TO BE OR NOT TO BE PLANAR ? */
00223   /*********************************/
00224
00225 PLANAR: PROCEDURE (N,M,SL,NARC,UU,LU,TYPE,FI,PRED,DUS,ERA) ;
00226
00227 DCL  (N,M,(SL,NARC,UU,LU,TYPE,FI,PRED,DUS,ERA)(*))  BIN FIXED(15,0) ;
00228 DCL  (FUFLICHOI,MERGETHIN,MERGETHICK,DELETION)  ENTRY ;
00229 DCL  1 BAT(1:N)                          BIN FIXED(15,0) INIT((N)0) ,
00230      1 STACK(1:N-1)                      BIN FIXED(15,0) ,
00231      1 TWUN ,
00232       2 (LT,RT,LB,RB)(1:M-N+1)           BIN FIXED(15,0) ,
00233       2 (K,KM1,DUM)                      BIN FIXED(15,0) ,
00234      1 SOL(1:M-N+1)                      BIN FIXED(15,0) ,
00235      1 (X,Y,P,V,NARCV,KK,SOLKK,SIDE)     BIN FIXED(15,0) ;
00236
00237 P,K=0 ; KM1=-1 ; DUM=M+1 ; KK,SOL(1),SOLKK=1 ; V=SL(1) ;
00238
00239 A: DO WHILE (V^=0);
00240      NARCV=NARC(V) ;
00241      IF NARCV<N
00242        THEN DO;/* NARCV: Tree Arc */
00243                P=P+1 ; STACK(P)=V ;
00244                IF BAT(LV(NARCV))=1 THEN DO; KK=KK+1 ; SOL(KK),SOLKK=K+1 ; END;
00245                V=SL(UV(NARCV)) ;
00246             END;
00247        ELSE DO;/* NARCV: Cotree Arc */
00248                KM1=K ; K=K+1 ; LT(K),LB(K)=NARCV ; RT(K),RB(K)=DUM ; SIDE=0 ;
00249                CALL FUFLICHOI (TWUN,PRED,SOLKK,UU,LU,LV(NARCV),NARCV,SIDE,FI) ;
00250                V=SL(V) ;
00251             END;
00252 END A;
00253
00254 IF P>0
00255   THEN DO;/* backing on the tree */
00256          V=STACK(P) ; P=P-1 ; NARCV=NARC(V) ; X=LV(NARCV) ; Y=UV(NARCV) ;
00257          CALL DELETION (TWUN,PRED,X,LU,FI,DUS) ;
00258          IF BAT(X)=0 THEN BAT(X)=1 ;/* 1st backtrack at X */
00259                      ELSE IF TYPE(NARCV)=1
00260                             THEN CALL MERGETHICK (TWUN,PRED,SOL,KK,SOLKK,UU,LU,
00261                                                   Y,NARCV,SIDE,FI,DUS,ERA) ;
00262                             ELSE CALL MERGETHIN (TWUN,PRED,SOL,KK,SOLKK,UU,LU,
00263                                                  Y,NARCV,SIDE,FI,DUS,ERA) ;
00264          V=SL(V) ; GOTO A ;
00265        END;
00266
00267 END PLANAR;
```

```
00268   /*************************************************************************/
00269   /* CHOICE PROCEDURE BETWEEN 'XFUSION' , 'HFUSION' & 'FLIPPING1' */
00270   /*************************************************************************/
00271
00272 FUFLICHOI: PROCEDURE (TWUN,PRED,SOLKK,UU,LU,Y,NARCU,SIDE,FI) ;
00273
00274 DCL  1 TWUN ,
00275       2 ((LT,RT,LB,RB)(*),K,KM1,DUM)            BIN FIXED(15,0) ,
00276      1 (SOLKK,Y,NARCU,SIDE,(UU,LU,FI,PRED)(*))  BIN FIXED(15,0) ;
00277 DCL  (XFUSION,HFUSION,FLIPPING1)  ENTRY ;
00278
00279 DO WHILE (KM1)=SOLKK);
00280   IF Y(LU(LT(KM1)))
00281     THEN IF Y(LU(RT(KM1))
00282            THEN CALL FLIPPING1 (TWUN,PRED,SOLKK,UU,LU,Y,NARCU,SIDE,FI) ;
00283            ELSE IF SIDE=0 THEN CALL XFUSION (TWUN,PRED,NARCU,SIDE) ;
00284                           ELSE CALL HFUSION (TWUN,PRED,NARCU) ;
00285     ELSE IF Y(LU(RT(KM1))
00286            THEN IF SIDE=0 THEN CALL HFUSION (TWUN,PRED,NARCU) ;
00287                           ELSE CALL XFUSION (TWUN,PRED,NARCU,SIDE) ;
00288            ELSE RETURN ;
00289
00290   IF PRED(NARCU)^=DUM THEN RETURN ;
00291 END;
00292
00293 END FUFLICHOI;
```

```
00294   /*************************************************************************/
00295   /* FUSION (LEFT/RIGHT) & (RIGHT/LEFT) BETWEEN LEVELS 'K' & 'KM1' */
00296   /*************************************************************************/
00297
00298 XFUSION: PROCEDURE (TWUN,PRED,NARCU,SIDE) ;
00299
00300 DCL  1 TWUN ,
00301       2 ((LT,RT,LB,RB)(*),K,KM1,DUM)  BIN FIXED(15,0) ,
00302      1 (NARCU,SIDE,PRED(*))           BIN FIXED(15,0) ;
00303
00304 IF RT(KM1)=DUM
00305   THEN DO; RB(KM1)=NARCU ; RT(KM1)=LT(K) ; PRED(NARCU)=DUM ;
00306            IF RB(K)^=DUM THEN DO; PRED(RB(K))=LT(KM1) ; LT(KM1)=RT(K) ; END;
00307        END;
00308   ELSE IF LT(KM1)=DUM
00309          THEN DO; LB(KM1)=NARCU ; LT(KM1)=RT(K) ; PRED(NARCU)=DUM ;
00310                   PRED(LB(K))=RT(KM1) ; RT(KM1)=LT(K) ;
00311                END;
00312          ELSE DO; PRED(LB(K))=RT(KM1) ; RT(KM1)=LT(K) ;
00313                   IF RB(K)^=DUM
00314                     THEN DO; PRED(RB(K))=LT(KM1) ; LT(KM1)=RT(K) ; END;
00315                END;
00316
00317 K=KM1 ; KM1=KM1-1 ; SIDE=1-SIDE ;
00318
00319 END XFUSION;



00320   /*************************************************************************/
00321   /* FUSION (LEFT/LEFT) & (RIGHT/RIGHT) BETWEEN LEVELS 'K' & 'KM1' */
00322   /*************************************************************************/
00323
00324 HFUSION: PROCEDURE (TWUN,PRED,NARCU) ;
00325
00326 DCL  1 TWUN ,
00327       2 ((LT,RT,LB,RB)(*),K,KM1,DUM)  BIN FIXED(15,0) ,
00328      1 (NARCU,PRED(*))                BIN FIXED(15,0) ;
00329
00330 IF LT(KM1)=DUM
00331   THEN DO; LB(KM1)=NARCU ; LT(KM1)=LT(K) ; PRED(NARCU)=DUM ;
00332            IF RB(K)^=DUM THEN DO; PRED(RB(K))=RT(KM1) ; RT(KM1)=RT(K) ; END;
00333        END;
00334   ELSE IF RT(KM1)=DUM
00335          THEN DO; RB(KM1)=NARCU ; RT(KM1)=RT(K) ; PRED(NARCU)=DUM ;
00336                   PRED(LB(K))=LT(KM1) ; LT(KM1)=LT(K) ;
00337                END;
00338          ELSE DO; PRED(LB(K))=LT(KM1) ; LT(KM1)=LT(K) ;
00339                   IF RB(K)^=DUM
00340                     THEN DO; PRED(RB(K))=RT(KM1) ; RT(KM1)=RT(K) ; END;
00341                END;
00342
00343 K=KM1 ; KM1=KM1-1 ;
00344
00345 END HFUSION;
```

```
00346    /*****************************************/
00347    /* FLIPPING OF ARCS ON THE LEVEL 'KM1' */
00348    /*****************************************/
00349
00350 FLIPPING1: PROCEDURE (TWUN,PRED,SOLKK,UU,LU,Y,NARCU,SIDE,FI) ;
00351
00352 DCL  1 TWUN ,                                          BIN FIXED(15,0) ,
00353       2 ((LT,RT,LB,RB)(*),K,KM1,DUM)                  BIN FIXED(15,0) ,
00354      1 (SOLKK,Y,NARCU,SIDE,(UU,LU,FI,PRED)(*))         BIN FIXED(15,0) ;
00355 DCL  TILT  LABEL  EXTERNAL ;
00356 DCL  (XFUSION,HFUSION)  ENTRY ;
00357 DCL  COVE  BIN FIXED(15,0) ;
00358
00359 COVE=LB(SOLKK) ;
00360 IF COVE=DUM THEN COVE=RB(SOLKK) ;
00361              ELSE IF RB(SOLKK)^=DUM THEN IF LU(RB(SOLKK))<LU(COVE)
00362                                          THEN COVE=RB(SOLKK) ;
00363
00364 IF LU(LT(KM1))=LU(RT(KM1))
00365   THEN IF LU(RT(KM1))<=LU(LB(KM1)) &
00366           ( LU(COVE)>=LU(RT(KM1)) ! UU(COVE)=UU(RT(KM1)) )
00367          THEN DO;/* Flipping of the Right-Top under the Left-Bottom */
00368                  A: FI(RT(KM1))=1-FI(RT(KM1)) ;
00369                     PRED(LB(KM1)),LB(KM1)=RT(KM1) ;
00370                     RT(KM1)=PRED(RT(KM1)) ; PRED(LB(KM1))=DUM ;
00371                     IF Y(LU(RT(KM1)) THEN GOTO A ;
00372                     IF SIDE=0 THEN CALL XFUSION (TWUN,PRED,NARCU,SIDE) ;
00373                               ELSE CALL HFUSION (TWUN,PRED,NARCU) ;
00374                     RETURN ;
00375               END;
00376          ELSE IF LU(LT(KM1))>LU(RT(KM1)) THEN GOTO TILT ;
00377
00378 IF LU(LT(KM1))<=LU(RB(KM1)) &
00379    ( LU(COVE)>=LU(LT(KM1)) ! UU(COVE)=UU(LT(KM1)) )
00380   THEN DO;/* Flipping of the Left-Top under the Right-Bottom */
00381           B: FI(LT(KM1))=1-FI(LT(KM1)) ;
00382              PRED(RB(KM1)),RB(KM1)=LT(KM1) ;
00383              LT(KM1)=PRED(LT(KM1)) ; PRED(RB(KM1))=DUM ;
00384              IF Y(LU(LT(KM1)) THEN GOTO B ;
00385              IF SIDE=0 THEN CALL HFUSION (TWUN,PRED,NARCU) ;
00386                        ELSE CALL XFUSION (TWUN,PRED,NARCU,SIDE) ;
00387        END;
00388   ELSE GOTO TILT ;
00389
00390 END FLIPPING1;
```

```
00391   /*************************************************************/
00392   /* FLIPPING OF ALL RIGHT-SIDE ARCS OF THE LEVEL 'LEV' */
00393   /* TO MAKE A LEFT LEV-STACK                                  */
00394   /*************************************************************/
00395
00396 FLIPPING2: PROCEDURE (TWUN,PRED,LEV,FI,DUS,ERA) ;
00397
00398 DCL  1 TWUN ,
00399       2 ((LT,RT,LB,RB)(*),K,KM1,DUM)  BIN FIXED(15,0) ,
00400      1 (LEV,(FI,PRED,DUS,ERA)(*))     BIN FIXED(15,0) ;
00401 DCL  RTLEV  BIN FIXED(15,0) ;
00402
00403 RTLEV=RT(LEV) ;
00404
00405 DO WHILE (RTLEV^=DUM);
00406   FI(RTLEV)=1-FI(RTLEV) ; DUS(RTLEV)=FI(RTLEV)+FI(LB(LEV)) ;
00407   RT(LEV)=PRED(RTLEV) ; ERA(RTLEV)=1 ; PRED(RTLEV)=LB(LEV) ; RTLEV=RT(LEV) ;
00408 END;
00409
00410 RB(LEV)=DUM ;
00411
00412 END FLIPPING2;
```

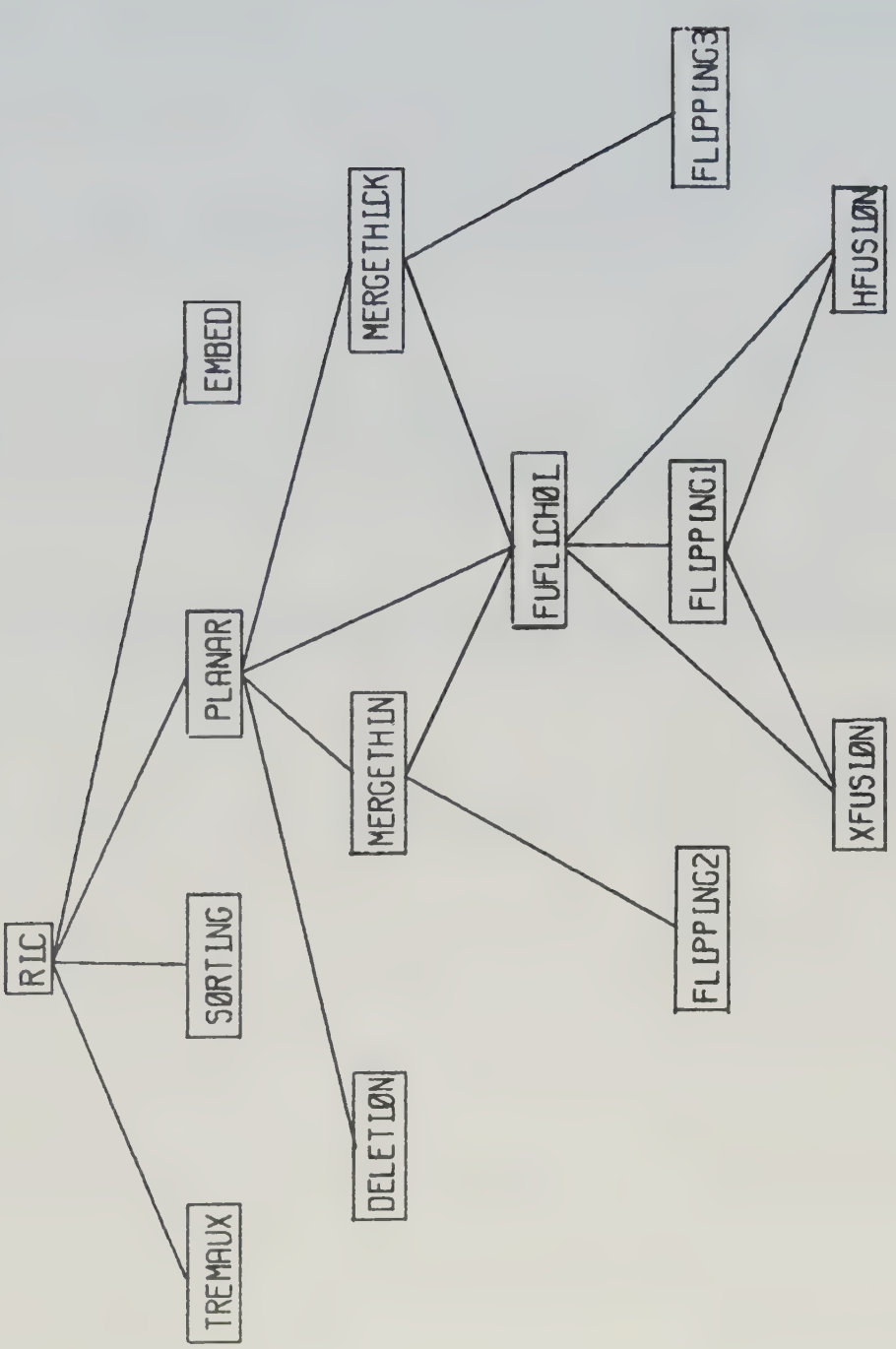
```
00413   /*****************************************************************/
00414   /* FLIPPING OF ALL LEFT OR RIGHT -SIDE ARCS OF THE LEVEL 'LEV'   */
00415   /* AND MAKE A LEFT LEV-STACK                                     */
00416   /*****************************************************************/
00417
00418 FLIPPING3: PROCEDURE (TWUN,PRED,LEV,UV,LV,FI,COVE,DUS,ERA) ;
00419
00420 DCL  1 TWUN ,
00421       2 ((LT,RT,LB,RB)(*),K,KM1,DUM)          BIN FIXED(15,0) ,
00422      1 ((UV,LV,FI,PRED,DUS,ERA)(*),LEV,COVE)  BIN FIXED(15,0) ;
00423 DCL  TILT  LABEL  EXTERNAL ;
00424 DCL  (RTLEV,LTLEV)  BIN FIXED(15,0) ;
00425
00426 IF LV(LT(LEV))=LV(RT(LEV))
00427   THEN IF LV(RT(LEV))<=LV(LB(LEV)) &
00428           ( LV(COVE)>=LV(RT(LEV)) ! UV(COVE)=UV(RT(LEV)) )
00429         THEN DO;/* Flipping of the Right-Top under the Left-Bottom */
00430                 A: RTLEV=RT(LEV) ; FI(RTLEV)=1-FI(RTLEV) ;
00431                    RT(LEV)=PRED(RTLEV) ;
00432                    IF LV(RTLEV)<LV(LB(LEV))
00433                      THEN DO; PRED(LB(LEV)),LB(LEV)=RTLEV ;
00434                               PRED(LB(LEV))=DUM ;
00435                           END;
00436                      ELSE DO; PRED(RTLEV)=LB(LEV) ; ERA(RTLEV)=2 ;
00437                               DUS(RTLEV)=FI(RTLEV)+FI(LB(LEV)) ;
00438                           END;
00439                    IF RT(LEV)^=DUM THEN GOTO A ;
00440                    RB(LEV)=DUM ; RETURN ;
00441              END;
00442         ELSE IF LV(LT(LEV))>LV(RT(LEV)) THEN GOTO TILT ;
00443
00444 IF LV(LT(LEV))<=LV(RB(LEV)) &
00445    ( LV(COVE)>=LV(LT(LEV)) ! UV(COVE)=UV(LT(LEV)) )
00446   THEN DO;/* Flipping of the Left-Top under the Right-Bottom */
00447           B: LTLEV=LT(LEV) ; FI(LTLEV)=1-FI(LTLEV) ;
00448              LT(LEV)=PRED(LTLEV) ;
00449              IF LV(LTLEV)<LV(RB(LEV))
00450                THEN DO; PRED(RB(LEV)),RB(LEV)=LTLEV ;
00451                         PRED(RB(LEV))=DUM ;
00452                     END;
00453                ELSE DO; PRED(LTLEV)=RB(LEV) ; ERA(LTLEV)=2 ;
00454                         DUS(LTLEV)=FI(LTLEV)+FI(RB(LEV)) ;
00455                     END;
00456              IF LT(LEV)^=DUM THEN GOTO B ;
00457              LT(LEV)=RT(LEV) ; LB(LEV)=RB(LEV) ; RT(LEV),RB(LEV)=DUM ;
00458       END;
00459   ELSE GOTO TILT ;
00460
00461 END FLIPPING3;
```

```
00462   /*********************************************/
00463   /* MERGE IN THE TWUN OF A THIN TREE BRANCH */
00464   /*********************************************/
00465
00466 MERGETHIN: PROCEDURE (TWUN,PRED,SOL,KK,SOLKK,UU,LU,Y,NARCU,SIDE,FI,DUS,ERA) ;
00467
00468 DCL  1 TWUN ,
00469       2 ((LT,RT,LB,RB)(*),K,KM1,DUM)                              BIN FIXED(15,0) ,
00470      1 (KK,SOLKK,Y,NARCU,SIDE,(SOL,UU,LU,FI,PRED,DUS,ERA)(*))     BIN FIXED(15,0) ;
00471 DCL  (FLIPPING2,FUFLICHOI)  ENTRY ;
00472 DCL  (LBK,LTK)  BIN FIXED(15,0) ;
00473
00474 IF RB(K)^=DUM
00475   THEN IF LB(K)=DUM
00476          THEN DO; LT(K)=RT(K) ; LB(K)=RB(K) ; RT(K),RB(K)=DUM ; END;
00477          ELSE CALL FLIPPING2 (TWUN,PRED,K,FI,DUS,ERA) ;
00478
00479 DO WHILE (K)SOLKK);
00480   IF RB(KM1)^=DUM
00481     THEN IF LB(KM1)=DUM
00482            THEN DO; PRED(LB(K))=RT(KM1) ; LT(KM1)=LT(K) ;
00483                     LB(KM1)=RB(KM1) ; RT(KM1),RB(KM1)=DUM ;
00484                 END;
00485            ELSE DO; CALL FLIPPING2 (TWUN,PRED,KM1,FI,DUS,ERA) ;
00486                     PRED(LB(K))=LT(KM1) ; LT(KM1)=LT(K) ;
00487                 END;
00488     ELSE DO; PRED(LB(K))=LT(KM1) ; LT(KM1)=LT(K) ; END;
00489
00490   K=KM1 ; KM1=KM1-1 ;
00491 END;
00492
00493 LBK=LB(K) ; LTK=LT(K) ;
00494 DO WHILE(LTK^=LBK);
00495   DUS(LTK)=FI(LTK)+FI(PRED(LTK)) ; ERA(LTK)=1 ; LTK=PRED(LTK) ;
00496 END;
00497 NARCU,LT(K)=LBK ; SIDE=0 ; Y=LU(NARCU) ;
00498 KK=KK-1 ; SOLKK=SOL(KK) ;
00499 CALL FUFLICHOI (TWUN,PRED,SOLKK,UU,LU,Y,NARCU,SIDE,FI) ;
00500
00501 END MERGETHIN;
```

```
00502   /*************************************************/
00503   /* MERGE IN THE TWUN OF A THICK TREE BRANCH */
00504   /*************************************************/
00505
00506 MERGETHICK: PROCEDURE (TWUN,PRED,SOL,KK,SOLKK,UU,LU,Y,NARCU,SIDE,FI,DUS,ERA) ;
00507
00508 DCL  1 TWUN ,                                                    BIN FIXED(15,0) ,
00509        2 ((LT,RT,LB,RB)(*),K,KM1,DUM)                           BIN FIXED(15,0) ,
00510      1 (KK,SOLKK,Y,NARCU,SIDE,(SOL,UU,LU,FI,PRED,DUS,ERA)(*))    BIN FIXED(15,0) ;
00511 DCL  (FLIPPING3,FUFLICHOI)  ENTRY ;
00512 DCL  COVE  BIN FIXED(15,0) ;
00513
00514 COVE=LB(SOLKK) ;
00515 IF COVE=DUM THEN COVE=RB(SOLKK) ;
00516              ELSE IF RB(SOLKK)^=DUM THEN IF LU(RB(SOLKK))<LU(COVE)
00517                                          THEN COVE=RB(SOLKK) ;
00518
00519 IF RB(K)^=DUM
00520   THEN IF LB(K)=DUM
00521          THEN DO; LT(K)=RT(K) ; LB(K)=RB(K) ; RT(K),RB(K)=DUM ; END;
00522          ELSE CALL FLIPPING3 (TWUN,PRED,K,UU,LU,FI,COVE,DUS,ERA) ;
00523
00524 DO WHILE (K)>SOLKK);
00525   IF RB(KM1)^=DUM
00526     THEN IF LB(KM1)=DUM
00527            THEN DO; PRED(LB(K))=RT(KM1) ; LT(KM1)=LT(K) ;
00528                     LB(KM1)=RB(KM1) ; RT(KM1),RB(KM1)=DUM ;
00529                 END;
00530            ELSE DO; CALL FLIPPING3 (TWUN,PRED,KM1,UU,LU,FI,COVE,DUS,ERA) ;
00531                     PRED(LB(K))=LT(KM1) ; LT(KM1)=LT(K) ;
00532                 END;
00533     ELSE DO; PRED(LB(K))=LT(KM1) ; LT(KM1)=LT(K) ; END;
00534
00535   K=KM1 ; KM1=KM1-1 ;
00536 END;
00537
00538 NARCU=LB(K) ; SIDE=0 ; Y=LU(NARCU) ;
00539 KK=KK-1 ; SOLKK=SOL(KK) ;
00540 CALL FUFLICHOI (TWUN,PRED,SOLKK,UU,LU,Y,NARCU,SIDE,FI) ;
00541
00542 END MERGETHICK;
```

```
00543    /*****************************************/
00544    /* DELETION OF ALL ARCS FROM THE TOP-TWUN */
00545    /* HAVING 'X' FOR EXTREMITY               */
00546    /*****************************************/
00547
00548 DELETION: PROCEDURE (TWUN,PRED,X,LV,FI,DUS) ;
00549
00550 DCL  1 TWUN ,
00551        2 ((LT,RT,LB,RB)(*),K,KM1,DUM)  BIN FIXED(15,0) ,
00552      1 ((LV,PRED,FI,DUS)(*),X)         BIN FIXED(15,0) ;
00553 DCL  (RTK,LTK,VAR)  BIN FIXED(15,0) ;
00554
00555 DO WHILE (K)0);
00556   IF LB(K)=DUM
00557     THEN DO; RTK=RT(K) ;
00558              DO WHILE (X=LV(RTK));
00559                DUS(RTK)=FI(RTK)+FI(PRED(RTK)) ;
00560                RT(K),RTK=PRED(RTK) ;
00561              END;
00562              IF RTK=DUM THEN DO; K=KM1 ; KM1=KM1-1 ; END;
00563                         ELSE RETURN ;
00564          END;
00565     ELSE DO; LTK=LT(K) ;
00566              DO WHILE (X=LV(LTK));
00567                DUS(LTK)=FI(LTK)+FI(PRED(LTK)) ;
00568                VAR=LTK ; LT(K),LTK=PRED(LTK) ;
00569              END;
00570              IF LTK=DUM
00571                THEN IF RB(K)=DUM
00572                       THEN DO; K=KM1 ; KM1=KM1-1 ; END;
00573                       ELSE DO; DUS(VAR)=DUS(VAR)+1 ;
00574                                PRED(VAR)=RB(K) ;
00575                                RTK=RT(K) ;
00576                                DO WHILE(X=LV(RTK));
00577                                  DUS(RTK)=FI(RTK)+FI(PRED(RTK)) ;
00578                                  RT(K),RTK=PRED(RTK) ;
00579                                END;
00580                                IF RTK=DUM THEN DO; K=KM1 ; KM1=KM1-1 ; END;
00581                                           ELSE DO; LB(K)=DUM ; RETURN ; END;
00582                            END;
00583                ELSE DO; IF RB(K)^=DUM
00584                           THEN DO; RTK=RT(K) ;
00585                                    DO WHILE(X=LV(RTK));
00586                                      DUS(RTK)=FI(RTK)+FI(PRED(RTK)) ;
00587                                      VAR=RTK ; RT(K),RTK=PRED(RTK) ;
00588                                    END;
00589                                    IF RTK=DUM THEN DO; DUS(VAR)=DUS(VAR)+1 ;
00590                                                        PRED(VAR)=LB(K) ;
00591                                                        RB(K)=DUM ;
00592                                                    END;
00593                                END;
00594                         RETURN ;
00595                     END;
00596          END;
00597 END;
00598
00599 END DELETION;
```

```
00600   /*************************************************/
00601   /* COMPUTE THE 'UPPER-SIDE',THE 'LOWER-SIDE' */
00602   /* AND THE 'LIB' OF EACH COTREE ARC          */
00603   /*************************************************/
00604
00605 EMBED: PROCEDURE (N,M,MP1,PRED,FI,DUS,ERA,US,LS,LIB) ;
00606
00607 DCL  ((PRED,FI,DUS,ERA,US,LS,LIB)(*),N,M,MP1)  BIN FIXED(15,0) ;
00608 DCL  (I,X,Y,L,P,V)     BIN FIXED(15,0) ,
00609      SOR(N:2*M-N+2)    BIN FIXED(15,0) INIT((2*M-2*N+3)0) ,
00610      EX(M+2:2*M-N+2)   BIN FIXED(15,0) ;
00611
00612 BEGIN; DCL  ID(N:MP1)  BIN FIXED(15,0) ;
00613 DO I=N TO MP1 BY 1; ID(I)=I ; END;
00614 I=MP1 ;
00615 DO Y=N TO M BY 1;
00616   X=PRED(Y) ; I=I+1 ; SOR(ID(X))=I ; ID(X)=I ; EX(I)=Y ;
00617 END; END;
00618
00619 BEGIN; DCL  STACK(1:M-N+2)  BIN FIXED(15,0) ;
00620 P=0 ; V=SOR(MP1) ;
00621
00622 A: DO WHILE(V^=0);
00623      Y=EX(V) ; X=PRED(Y) ; P=P+1 ; STACK(P)=V ;
00624      IF P=1 THEN DO; L=L+1 ;
00625                      US(Y)=0 ;
00626                      LS(Y)=MOD(FI(Y),2) ;
00627                  END;
00628             ELSE DO; US(Y)=MOD(US(X)+DUS(Y),2) ;
00629                      IF ERA(Y)^=0 THEN FI(Y)=FI(X)+DUS(Y) ;
00630                      LS(Y)=MOD(US(Y)+FI(Y),2) ;
00631                  END;
00632      LIB(Y)=L ; V=SOR(Y) ;
00633 END A;
00634
00635 IF P>0 THEN DO; V=SOR(STACK(P)) ; P=P-1 ; GOTO A ; END;
00636
00637 END; END EMBED;
```



SCHEMA SYNOPTIQUE

III/ 6 EXECUTIONS

- Graphe de Martin GARDNER

  Nous laissons au lecteur le soin de le dessiner selon les résultats fournis par le programme ...

- Graphe n°5

  L'arbre est ici un chemin hamiltonien. Il y a 2 familles indépendantes d'arcs de coarbre.

- Graphe n°36

  Ici , tous les arcs du coarbre sont dépendants.

- Graphe n°40

  = blé4 de P. ROSENSTIEHL

- Graphe n°45

- Graphe n°52

1:	110	324
2:	1	2
3:	1	3
4:	1	4
5:	1	5
6:	1	6
7:	1	7
8:	1	8
9:	1	9
10:	1	10
11:	1	11
12:	1	20
13:	1	73
14:	1	109
15:	1	110
16:	2	3
17:	2	12
18:	2	57
19:	2	102
20:	2	103
21:	2	104
22:	2	105
23:	2	106
24:	2	110
25:	3	4
26:	3	12
27:	3	13
28:	4	5
29:	4	13
30:	4	14
31:	5	6
32:	5	14
33:	5	15
34:	6	7
35:	6	15
36:	6	16
37:	7	8
38:	7	16
39:	7	17
40:	8	9
41:	8	17
42:	8	18
43:	9	10
44:	9	18
45:	9	19
46:	10	11
47:	10	19
48:	10	20
49:	11	20
50:	12	13
51:	12	21
52:	12	57
53:	12	58
54:	13	14
55:	13	21
59:	14	23
60:	15	16
61:	15	23
62:	15	24
63:	16	17
64:	16	24
65:	16	25
66:	17	18
67:	17	25
68:	17	26
69:	18	19
70:	18	26
71:	18	27
72:	19	20
73:	19	27
74:	19	28
75:	20	28
76:	20	73
77:	21	22
78:	21	29
79:	21	58
80:	21	59
81:	22	23
82:	22	29
83:	22	30
84:	23	24
85:	22	30
86:	23	31
87:	24	25
88:	24	31
89:	24	32
90:	25	26
91:	25	32
92:	25	33
93:	26	27
94:	26	33
95:	26	34
96:	27	28
97:	27	34
98:	27	35
99:	28	35
100:	28	72
101:	28	73
102:	29	30
103:	29	36
104:	29	59
105:	29	60
106:	30	31
107:	30	36
108:	30	37
109:	31	32
110:	31	37
111:	31	38
112:	32	33
113:	32	38
114:	32	39

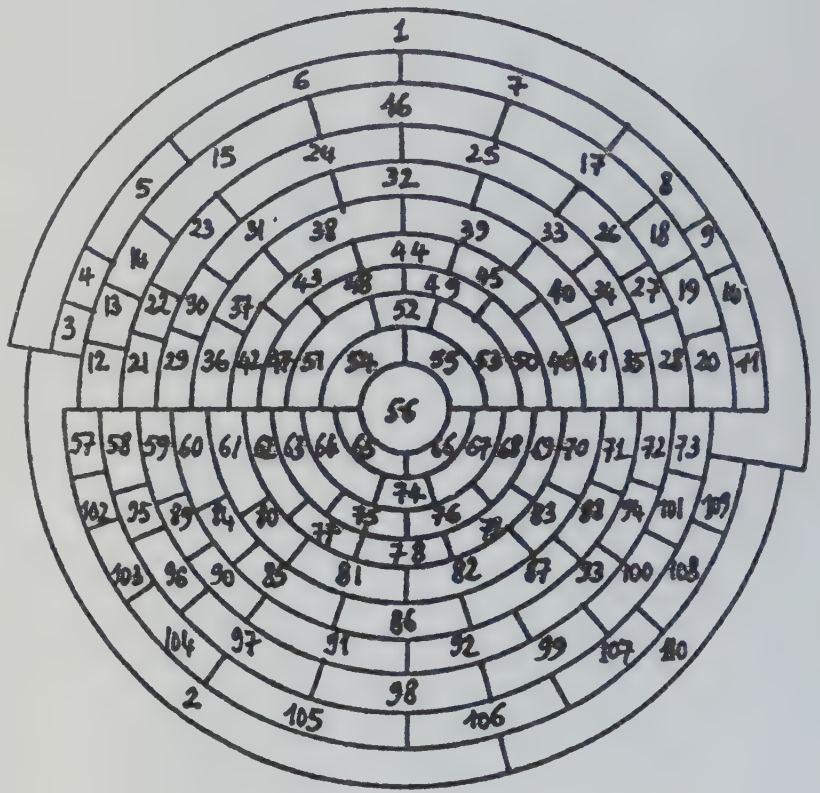
115:	33	34
116:	33	39
117:	33	40
118:	34	35
119:	34	40
120:	34	41
121:	35	41
122:	35	71
123:	35	72
124:	36	37
125:	36	42
126:	36	60
127:	36	61
128:	37	38
129:	37	42
130:	37	43
131:	38	39
132:	38	43
133:	38	44
134:	39	40
135:	39	44
136:	39	45
137:	40	41
138:	40	45
139:	40	46
140:	41	46
141:	41	70
142:	41	71
143:	42	43
144:	42	47
145:	42	61
146:	42	62
147:	43	44
148:	43	47
149:	43	48
150:	44	45
151:	44	48
152:	44	49
153:	45	46
154:	45	49
155:	45	50
156:	46	50
157:	46	69
158:	46	70
159:	47	48
160:	47	51
161:	47	62
162:	47	63
163:	48	49
164:	48	51
165:	48	52
166:	49	50
167:	49	52
168:	49	53
169:	50	53
170:	50	68
171:	50	69
174:	51	63
175:	51	64
176:	52	53
177:	52	54
178:	52	55
179:	53	55
180:	53	67
181:	53	68
182:	54	55
183:	54	56
184:	54	64
185:	54	65
186:	55	56
187:	55	66
188:	55	67
189:	56	65
190:	56	66
191:	57	58
192:	57	102
193:	58	59
194:	58	95
195:	58	102
196:	59	60
197:	59	89
198:	59	95
199:	60	61
200:	60	84
201:	60	89
202:	61	62
203:	61	80
204:	61	84
205:	62	63
206:	62	77
207:	62	80
208:	63	64
209:	63	75
210:	63	77
211:	64	65
212:	64	74
213:	64	75
214:	65	66
215:	65	74
216:	66	67
217:	66	74
218:	67	68
219:	67	74
220:	67	76
221:	68	69
222:	68	76
223:	68	79
224:	69	70
225:	69	79
226:	69	83
227:	70	71
228:	70	83
229:	70	88
230:	71	72
233:	72	73
234:	72	94
235:	72	101
236:	73	101
237:	73	109
238:	74	75
239:	74	76
240:	75	76
241:	75	77
242:	75	78
243:	76	78
244:	76	79
245:	77	78
246:	77	80
247:	77	81
248:	78	79
249:	78	81
250:	78	82
251:	79	82
252:	79	83
253:	80	81
254:	80	84
255:	80	85
256:	81	82
257:	81	85
258:	81	86
259:	82	83
260:	82	86
261:	82	87
262:	83	87
263:	83	88
264:	84	85
265:	84	89
266:	84	90
267:	85	86
268:	85	90
269:	85	91
270:	86	87
271:	86	91
272:	86	92
273:	87	88
274:	87	92
275:	87	93
276:	88	93
277:	88	94
278:	89	90
279:	89	95
280:	89	96
281:	90	91
282:	90	96
283:	90	97
284:	91	92
285:	91	97
286:	91	98
287:	92	93
288:	92	98
289:	92	99
292:	93	100
293:	94	100
294:	94	101
295:	95	96
296:	95	102
297:	95	103
298:	96	97
299:	96	103
300:	96	104
301:	97	98
302:	97	104
303:	97	105
304:	98	99
305:	98	105
306:	98	106
307:	99	100
308:	99	106
309:	99	107
310:	100	101
311:	100	107
312:	100	108
313:	101	108
314:	101	109
315:	102	103
316:	103	104
317:	104	105
318:	105	106
319:	106	107
320:	106	110
321:	107	108
322:	107	110
323:	108	109
324:	108	110
325:	109	110

NUMBER OF VERTICES= 110 NUMBER OF EDGES= 324

*** TREE ARCS ****

(X , Y)

(1, 2)
(2, 3)
(3, 4)
(4, 5)
(5, 6)
(6, 7)
(7, 8)
(8, 9)
(9, 10)
(10, 11)
(11, 20)
(20, 19)
(19, 18)
(18, 17)
(17, 16)
(16, 15)
(15, 14)
(14, 13)
(13, 12)
(12, 21)
(21, 22)
(22, 23)
(23, 24)
(24, 25)
(25, 26)
(26, 27)
(27, 28)
(28, 35)
(35, 34)
(34, 33)
(33, 32)
(32, 31)
(31, 30)
(30, 29)
(29, 36)
(36, 37)
(37, 38)
(38, 39)
(39, 40)
(40, 41)
(41, 46)
(46, 45)
(45, 44)
(44, 43)



(43, 42)
(42, 47)
(47, 48)
(48, 49)
(49, 50)
(50, 53)
(53, 52)
(52, 51)
(51, 54)
(54, 55)
(55, 56)
(56, 65)
(65, 64)
(64, 63)
(63, 62)
(62, 61)
(61, 60)
(60, 59)
(59, 58)
(58, 57)
(57,102)
(102, 95)
(95, 89)
(89, 84)
(84, 80)
(80, 77)
(77, 75)
(75, 74)
(74, 66)
(66, 67)
(67, 68)
(68, 69)
(69, 70)
(70, 71)
(71, 72)
(72, 73)
(73,101)
(101, 94)
(94, 88)
(88, 83)
(83, 79)
(79, 76)
(76, 78)
(78, 81)
(81, 82)
(82, 86)
(86, 85)
(85, 90)
(90, 91)
(91, 92)
(92, 87)
(87, 93)
(93, 99)
(99, 98)

(104,105)
(105,106)
(106,107)
(107,100)
(100,108)
(108,109)
(109,110)

**** COTREE ARCS ****

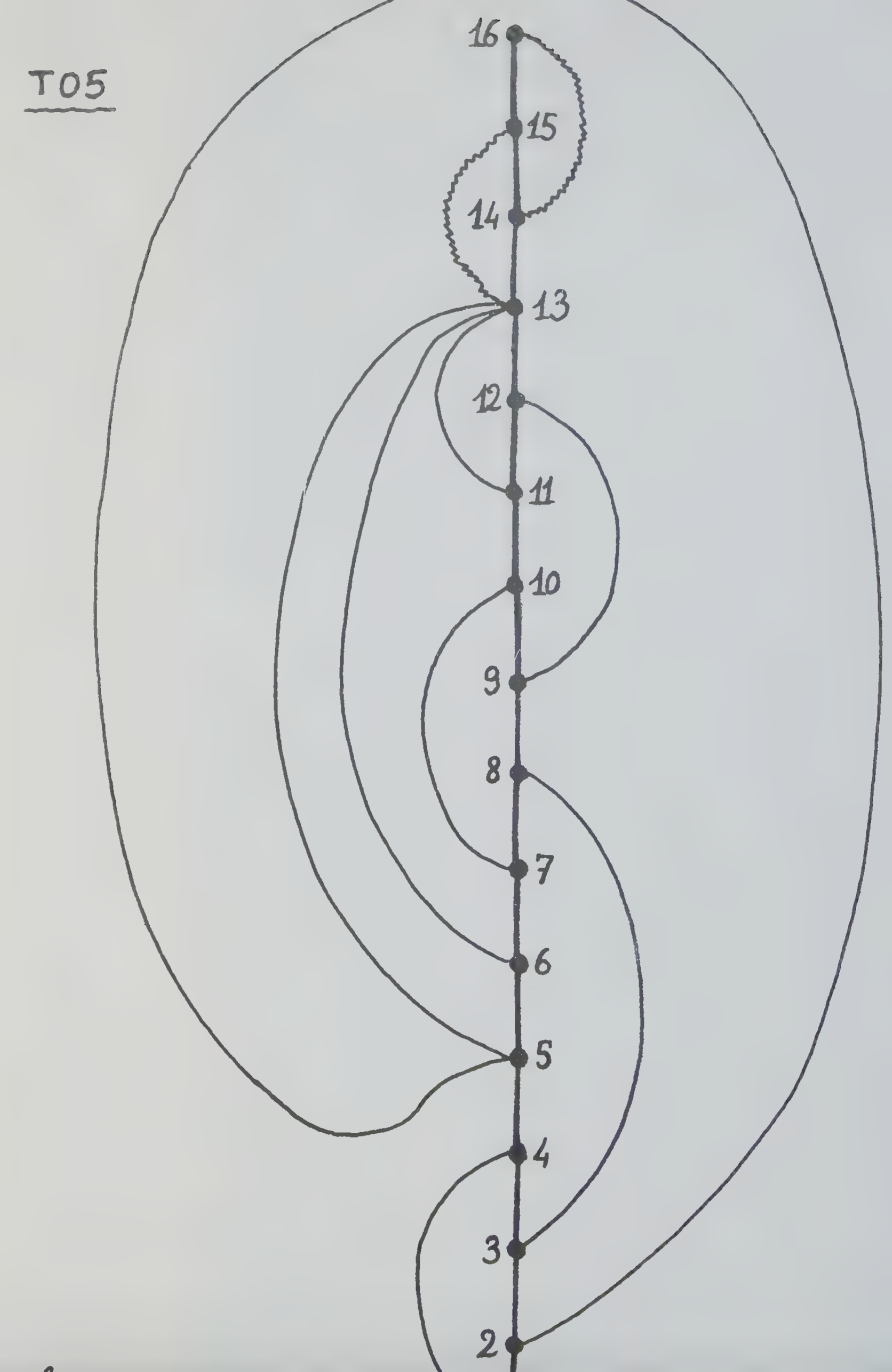
(X , Y)	US	LS	LIB
(3, 1)	0	0	1
(4, 1)	0	0	1
(5, 1)	0	0	1
(6, 1)	0	0	1
(7, 1)	0	0	1
(8, 1)	0	0	1
(9, 1)	0	0	1
(10, 1)	0	0	1
(11, 1)	0	0	1
(20, 1)	0	0	1
(20, 10)	1	1	1
(19, 9)	1	1	1
(19, 10)	1	1	1
(18, 8)	1	1	1
(18, 9)	1	1	1
(17, 7)	1	1	1
(17, 8)	1	1	1
(16, 6)	1	1	1
(16, 7)	1	1	1
(15, 5)	1	1	1
(15, 6)	1	1	1
(14, 4)	1	1	1
(14, 5)	1	1	1
(13, 3)	1	1	1
(13, 4)	1	1	1
(12, 2)	1	1	1
(12, 3)	1	1	1
(21, 13)	0	0	1
(22, 13)	0	0	1
(22, 14)	0	0	1
(23, 14)	0	0	1
(23, 15)	0	0	1
(24, 15)	0	0	1
(24, 16)	0	0	1
(25, 16)	0	0	1
(25, 17)	0	0	1
(26, 17)	0	0	1
(26, 18)	0	0	1
(27, 18)	0	0	1
(27, 19)	0	0	1
(28, 19)	0	0	1

(35, 27)	1	1	1
(34, 26)	1	1	1
(34, 27)	1	1	1
(33, 25)	1	1	1
(33, 26)	1	1	1
(32, 24)	1	1	1
(32, 25)	1	1	1
(31, 23)	1	1	1
(31, 24)	1	1	1
(30, 22)	1	1	1
(30, 22)	1	1	1
(29, 21)	1	1	1
(29, 22)	1	1	1
(36, 30)	0	0	1
(37, 30)	0	0	1
(37, 31)	0	0	1
(38, 31)	0	0	1
(38, 32)	0	0	1
(39, 32)	0	0	1
(39, 33)	0	0	1
(40, 33)	0	0	1
(40, 34)	0	0	1
(41, 34)	0	0	1
(41, 35)	0	0	1
(46, 40)	1	1	1
(45, 39)	1	1	1
(45, 40)	1	1	1
(44, 38)	1	1	1
(44, 39)	1	1	1
(43, 37)	1	1	1
(43, 38)	1	1	1
(42, 36)	1	1	1
(42, 37)	1	1	1
(47, 43)	0	0	1
(48, 43)	0	0	1
(48, 44)	0	0	1
(49, 44)	0	0	1
(49, 45)	0	0	1
(50, 45)	0	0	1
(50, 46)	0	0	1
(53, 49)	1	1	1
(52, 48)	1	1	1
(52, 49)	1	1	1
(51, 47)	1	1	1
(51, 48)	1	1	1
(54, 52)	0	0	1
(55, 52)	0	0	1
(55, 53)	0	0	1
(56, 54)	1	1	1
(65, 54)	1	1	1
(64, 51)	1	1	1
(64, 54)	1	1	1
(63, 47)	1	1	1
(63, 51)	1	1	1

(58, 12)	1	1	1	(82, 78)	0	0	1
(58, 21)	1	1	1	(82, 79)	0	0	1
(57, 2)	1	1	1	(82, 83)	0	0	1
(57, 12)	1	1	1	(86, 81)	1	1	1
(102, 2)	1	1	1	(85, 80)	1	1	1
(102, 58)	0	0	1	(85, 81)	1	1	1
(95, 58)	0	0	1	(85, 84)	1	1	1
(95, 59)	0	0	1	(90, 84)	1	1	1
(89, 59)	0	0	1	(90, 89)	1	1	1
(89, 60)	0	0	1	(91, 85)	0	0	1
(84, 60)	0	0	1	(91, 86)	0	0	1
(84, 61)	0	0	1	(92, 86)	0	0	1
(80, 61)	0	0	1	(87, 82)	0	0	1
(80, 62)	0	0	1	(87, 83)	0	0	1
(77, 62)	0	0	1	(87, 86)	0	0	1
(77, 63)	0	0	1	(87, 88)	0	0	1
(75, 63)	0	0	1	(93, 88)	0	0	1
(75, 64)	0	0	1	(93, 92)	1	1	1
(74, 64)	0	0	1	(93, 94)	0	0	1
(74, 65)	0	0	1	(99, 92)	1	1	1
(66, 55)	0	0	1	(98, 91)	1	1	1
(66, 56)	0	0	1	(98, 92)	1	1	1
(66, 65)	0	0	1	(97, 90)	1	1	1
(67, 53)	0	0	1	(97, 91)	1	1	1
(67, 55)	0	0	1	(96, 89)	1	1	1
(68, 50)	0	0	1	(96, 90)	1	1	1
(68, 53)	0	0	1	(96, 95)	1	1	1
(69, 46)	0	0	1	(103, 2)	1	1	1
(69, 50)	0	0	1	(103, 95)	1	1	1
(70, 41)	0	0	1	(103,102)	1	1	1
(70, 46)	0	0	1	(104, 2)	1	1	1
(71, 35)	0	0	1	(104, 96)	0	0	1
(71, 41)	0	0	1	(104, 97)	0	0	1
(72, 28)	0	0	1	(105, 2)	1	1	1
(72, 35)	0	0	1	(105, 97)	0	0	1
(73, 1)	0	0	1	(105, 98)	0	0	1
(73, 20)	0	0	1	(106, 2)	1	1	1
(73, 28)	0	0	1	(106, 98)	0	0	1
(101, 72)	1	1	1	(106, 99)	0	0	1
(94, 71)	1	1	1	(107, 99)	0	0	1
(94, 72)	1	1	1	(100, 93)	0	0	1
(88, 70)	1	1	1	(100, 94)	0	0	1
(88, 71)	1	1	1	(100, 99)	0	0	1
(83, 69)	1	1	1	(100,101)	0	0	1
(83, 70)	1	1	1	(108,101)	0	0	1
(79, 68)	1	1	1	(108,107)	1	1	1
(79, 69)	1	1	1	(109, 1)	0	0	1
(76, 67)	1	1	1	(109, 73)	0	0	1
(76, 68)	1	1	1	(109,101)	0	0	1
(76, 74)	1	1	1	(110, 1)	0	0	2
(76, 75)	1	1	1	(110, 2)	1	1	1
(78, 75)	1	1	1	(110,106)	1	1	1
(78, 77)	1	1	1	(110,107)	1	1	1
(78, 79)	0	0	1	(110,108)	1	1	1
				(67, 74)	1	1	1

****** F I C H I E R : TAL:RIC:GRAF C L E : GRAF-T05

1:	16	25
2:	1	2
3:	2	3
4:	3	4
5:	4	5
6:	5	6
7:	6	7
8:	7	8
9:	8	9
10:	9	10
11:	10	11
12:	11	12
13:	12	13
14:	13	14
15:	14	15
16:	15	16
17:	16	14
18:	15	13
19:	13	5
20:	13	6
21:	13	11
22:	12	9
23:	10	7
24:	8	3
25:	5	2
26:	4	1



NUMBER OF VERTICES: 16 NUMBER OF EDGES: 25

**** TREE ARCS ****

(X , Y)
(1, 2)
(2, 3)
(3, 4)
(4, 5)
(5, 6)
(6, 7)
(7, 8)
(8, 9)
(9, 10)
(10, 11)
(11, 12)
(12, 13)
(13, 14)
(14, 15)
(15, 16)

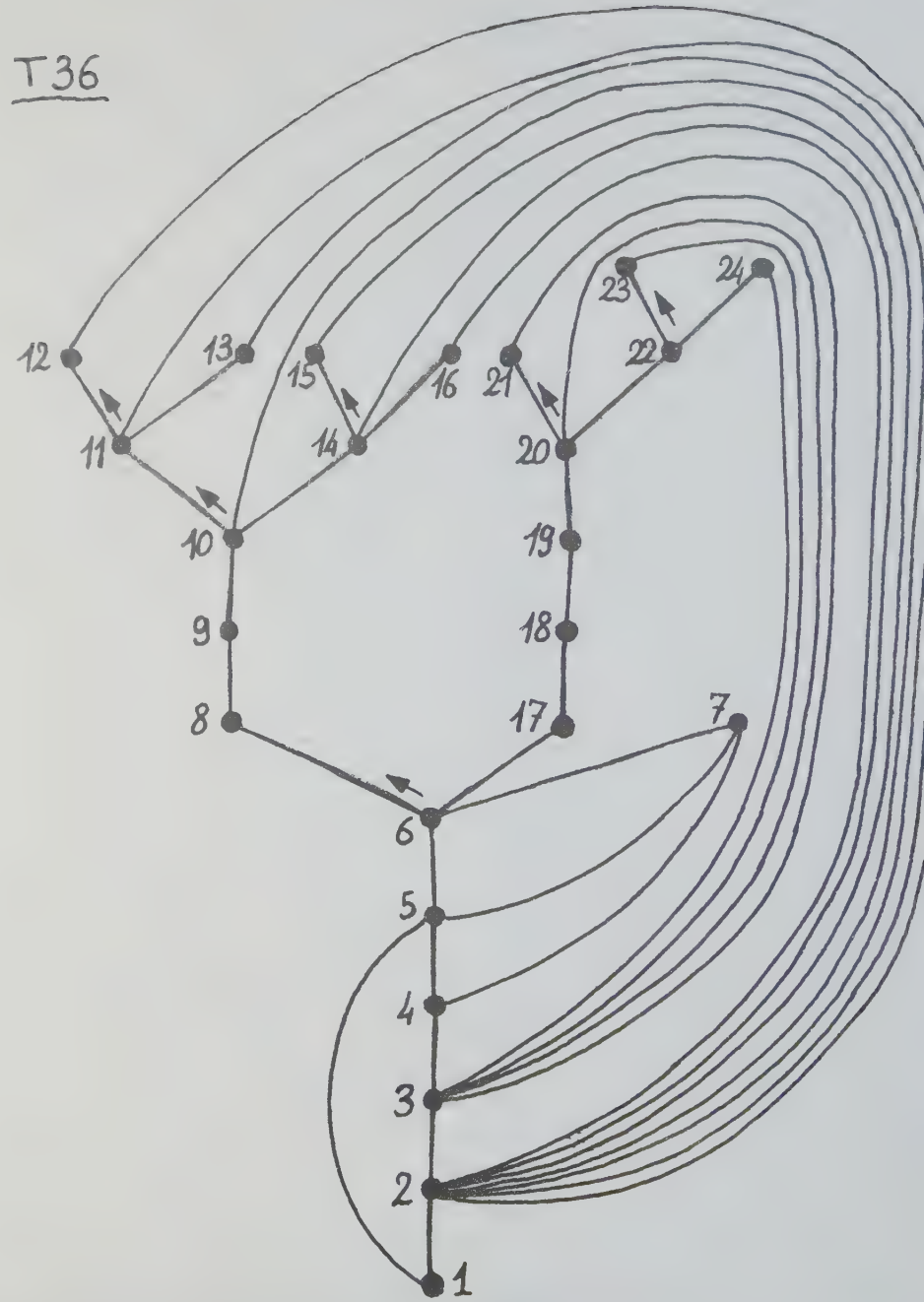
**** COTREE ARCS ****

(X , Y)	US	LS	LIB
(16, 14)	1	1	1
(15, 13)	0	0	1
(13, 5)	0	0	2
(13, 6)	0	0	2
(13, 11)	0	0	2
(12, 9)	1	1	2
(10, 7)	0	0	2
(8, 3)	1	1	2
(5, 2)	0	1	2
(4, 1)	0	0	2

** BRAVO | THIS GRAPH IS PLANAR **

****** F I C H I E R : TAL:RIC:GRAF C L E : GRAF-T36

1:	24	37
2:	1	2
3:	2	3
4:	3	4
5:	4	5
6:	5	6
7:	6	7
8:	6	8
9:	8	9
10:	9	10
11:	10	11
12:	11	12
13:	11	13
14:	10	14
15:	14	15
16:	14	16
17:	6	17
18:	17	18
19:	18	19
20:	19	20
21:	20	21
22:	20	22
23:	22	23
24:	22	24
25:	4	7
26:	5	7
27:	2	12
28:	2	13
29:	2	11
30:	2	15
31:	2	16
32:	2	14
33:	2	10
34:	3	21
35:	3	23
36:	3	24
37:	3	20
38:	1	5



NUMBER OF VERTICES: 24 NUMBER OF EDGES: 37

**** TREE ARCS ****

(X , Y)
(1, 2)
(2, 3)
(3, 4)
(4, 5)
(5, 6)
(6, 7)
(6, 8)
(8, 9)
(9, 10)
(10, 11)
(11, 12)
(11, 13)
(10, 14)
(14, 15)
(14, 16)
(6, 17)
(17, 18)
(18, 19)
(19, 20)
(20, 21)
(20, 22)
(22, 23)
(22, 24)

**** COTREE ARCS ****

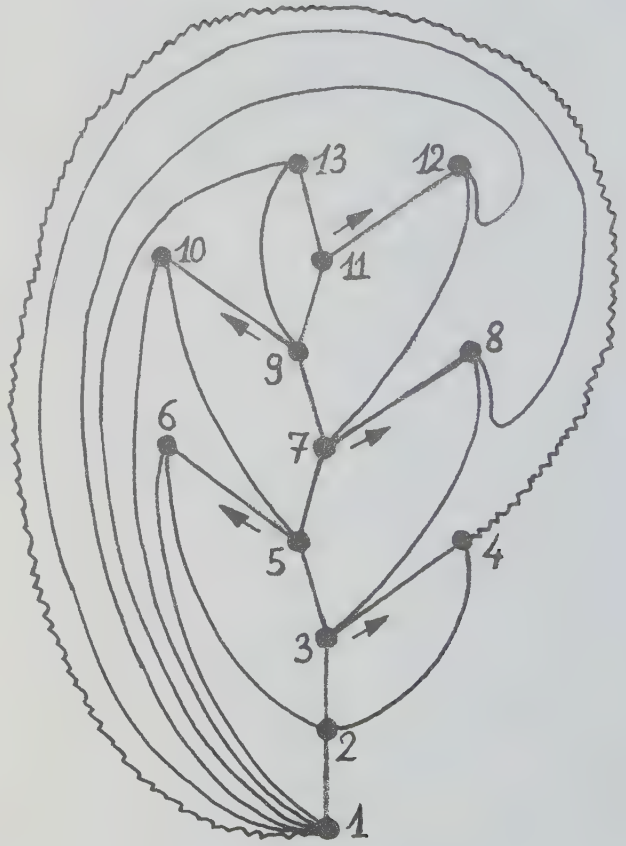
(X , Y)	US	LS	LIB
(7, 4)	1	1	1
(7, 5)	1	1	1
(12, 2)	1	1	1
(13, 2)	1	1	1
(11, 2)	1	1	1
(15, 2)	1	1	1
(16, 2)	1	1	1
(14, 2)	1	1	1
(10, 2)	1	1	1
(21, 3)	1	1	1
(23, 3)	1	1	1
(24, 3)	1	1	1
(20, 3)	1	1	1

** BRAVO | THIS GRAPH IS PLANAR **

****** F I C H I E R : TAL:RIC:GRAF C L E : GRAF-T40

1:	13	24
2:	1	2
3:	2	3
4:	3	4
5:	3	5
6:	5	6
7:	5	7
8:	7	8
9:	7	9
10:	9	10
11:	9	11
12:	11	12
13:	11	13
14:	1	13
15:	9	13
16:	1	12
17:	7	12
18:	1	10
19:	5	10
20:	1	8
21:	3	8
22:	1	6
23:	2	6
24:	1	4
25:	2	4

T40



NUMBER OF VERTICES: 13 NUMBER OF EDGES: 24

**** TREE ARCS ****

(X , Y)
(1, 2)
(2, 3)
(3, 4)
(3, 5)
(5, 6)
(5, 7)
(7, 8)
(7, 9)
(9, 10)
(9, 11)
(11, 12)
(11, 13)

**** COTREE ARCS ****

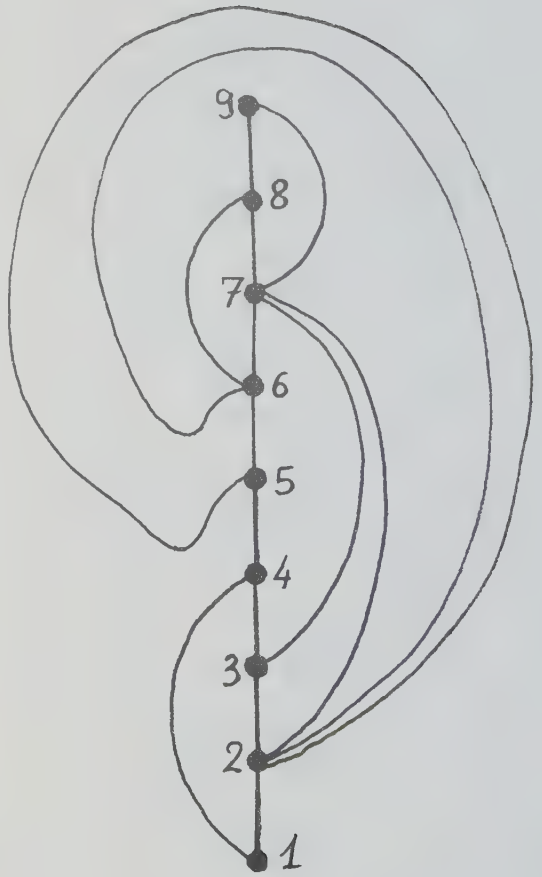
(X , Y)	US	LS	LIB
(4, 1)	0	0	1
(4, 2)	1	1	2
(6, 1)	0	0	2
(6, 2)	0	0	2
(8, 1)	1	0	2
(8, 3)	1	1	2
(10, 1)	0	0	2
(10, 5)	0	0	2
(12, 1)	1	0	2
(12, 7)	1	1	2
(13, 1)	0	0	2
(13, 9)	0	0	2

** BRAVO | THIS GRAPH IS PLANAR **

****** F I C H I E R : TAL:RIC:GRAF C L E : GRAF-T45

1:	9	15
2:	1	2
3:	2	3
4:	3	4
5:	4	5
6:	5	6
7:	6	7
8:	7	8
9:	8	9
10:	7	9
11:	6	8
12:	2	7
13:	3	7
14:	2	6
15:	2	5
16:	1	4

T45



NUMBER OF VERTICES: 9 NUMBER OF EDGES: 15

**** TREE ARCS ****

(X , Y)
(1, 2)
(2, 3)
(3, 4)
(4, 5)
(5, 6)
(6, 7)
(7, 8)
(8, 9)

**** COTREE ARCS ****

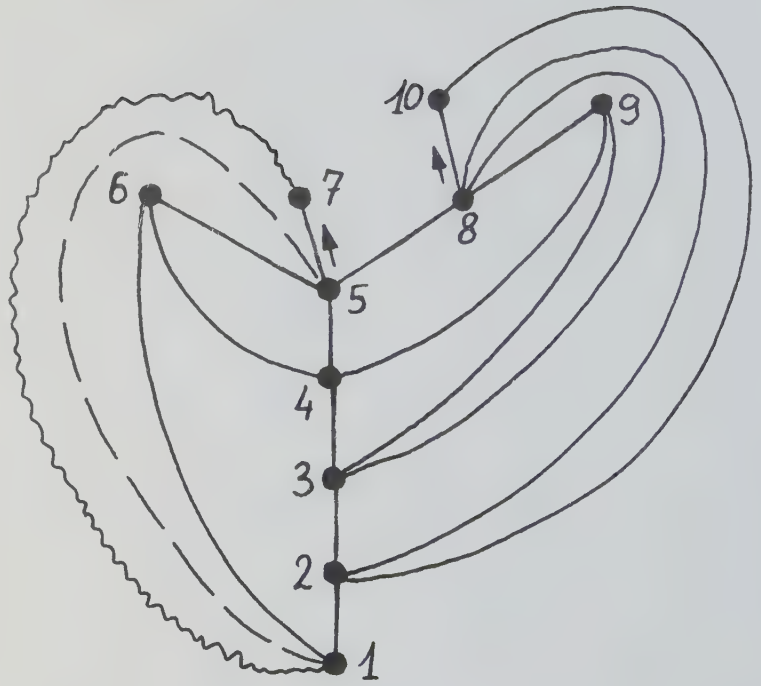
(X , Y)	US	LS	LIB
(9, 7)	1	1	1
(8, 6)	0	0	1
(7, 2)	1	1	1
(7, 3)	1	1	1
(6, 2)	0	1	1
(5, 2)	0	1	1
(4, 1)	0	0	1

** BRAVO | THIS GRAPH IS PLANAR **

****** F I C H I E R : TAL:RIC:GRAF C L E : GRAF-T52

1:	10	18
2:	1	2
3:	2	3
4:	3	4
5:	4	5
6:	5	6
7:	5	7
8:	5	8
9:	8	9
10:	8	10
11:	1	6
12:	4	6
13:	1	7
14:	3	9
15:	4	9
16:	2	10
17:	2	8
18:	3	8
19:	1	5

T52



NUMBER OF VERTICES: 10 NUMBER OF EDGES: 18

**** TREE ARCS ****

(X , Y)
(1, 2)
(2, 3)
(3, 4)
(4, 5)
(5, 6)
(5, 7)
(5, 8)
(8, 9)
(8, 10)

**** COTREE ARCS ****

(X , Y)	US	LS	LIB
(6, 1)	0	0	1
(6, 4)	0	0	1
(7, 1)	0	0	2
(9, 3)	1	1	1
(9, 4)	1	1	1
(10, 2)	1	1	1
(8, 2)	1	1	1
(8, 3)	1	1	1
(5, 1)	0	0	3

** BRAVO ! THIS GRAPH IS PLANAR **

Nous indiquons que la preuve de l'algorithme "Gauche-Droite" se trouvera dans :

"A new linear algorithm for planarity testing" ,
by H. de FRAYSSEIX , M. HABIB , J.P. RICHARD , P. ROSENSTIEHL .
(en préparation)

CHAPITRE III

ANALYSE DU PROGRAMME

I/ INTRODUCTION

Dans ce troisième chapitre , nous allons analyser le programme (version C1 présentée au précédent chapitre) en espace mémoire et en temps de calcul , afin de démontrer sa linéarité en "N" et "M" dans les deux cas.

II/ ANALYSE EN PLACE MEMOIRE

1) GENERALITES

Nous allons effectuer ce calcul en partant des procédures terminales (c-à-d. n'appelant aucune autre procédure ; voir le schéma synoptique p.73) et en remontant les niveaux d'appel jusqu'au programme principal.

Le comptage sera exprimé en "mot machine". Nous confondrons volontairement les variables en mots , demi-mots ou booléennes (cette différentiation reviendrait simplement à modifier les constantes du polynôme).

Définissons une fonction espace "&" , telle que & (TRUC) , "TRUC" étant une procédure , représente l'espace mémoire propre à cette procédure , ne comptant pas l'espace mémoire de ses paramètres.

Nous donnerons comme taille du programme la valeur maximale de l'encombrement mémoire de la procédure principale "RIC" :

$$\Rightarrow \quad \& (RIC) = \& (\text{version C1})$$

2) PROCEDURES "CONSTANTES"

Commençons notre analyse par les huit procédures qui nécessitent un espace mémoire constant au cours de leur(s) exécution(s).

On peut facilement trouver les valeurs suivantes :

& (TREMAUX) = 4N+9

& (SORTING) = 2M+11N+4

& (FLIPPING1) = 1

& (FLIPPING2) = 1

& (FLIPPING3) = 2

& (DELETION) = 3

& (XFUSION) = 0

& (HFUSION) = 0

3) PROCEDURES "VARIABLES"

Les procédures suivantes occupent un espace mémoire variable en fonction du temps. Rappelons que le PL1 est un langage de programmation possédant la notion de "blocs". Nous donnerons à chaque fois le numéro du bloc (≡ LEV du PL1) , l'espace mémoire interne à ce bloc et la somme d'espace occupé à l'instant "t".

Le symbole "(*)" signifie que l'appel du sous-programme n'est pas systématique.

3-1) FUFLICHOI

LEV	Place	Σ_t
(1)	0	0
Call FLIPPING1 (*)	1	1
Call XFUSION (*)	0	0
Call HFUSION (*)	0	0

$\Rightarrow$ & (FUFLICHOI) $\leqslant 1$

3-2) MERGETHIN

LEV	Place	Σ_t
(1)	2	2
Call FLIPPING2 (*)	1	3
Call FUFLICHOI	$\leqslant 1$	$\leqslant 3$

$\Rightarrow$ & (MERGETHIN) $\leqslant 3$

3-3) MERGETHICK

LEV	Place	Σ_t
(1)	1	1
Call FLIPPING3 (*)	2	3
Call FUFLICHOI	$\leqslant 1$	$\leqslant 2$

$\Rightarrow$ & (MERGETHICK) $\leqslant 3$

3-4) PLANAR

LEV	Place	Σ_t
(1)	5M-3N+15	5M-3N+15
Call FUFLICHOI (*)	$\leqslant 1$	$\leqslant$ 5M-3N+16
Call DELETION	3	5M-3N+18
Call MERGETHICK (*)	$\leqslant 3$	$\leqslant$ 5M-3N+18
Call MERGETHIN (*)	$\leqslant 3$	$\leqslant$ 5M-3N+18

$\Rightarrow$ & (PLANAR) = 5M-3N+18

3-5) EMBED

LEV	Place	Σ_t
(1)	3M-3N+10	3M-3N+10
(2)	M-N+2	4M-4N+12
(2)	M-N+2	4M-4N+12

$\Rightarrow$ & (EMBED) = 4M-4N+12

4) PROCEDURE PRINCIPALE

Nous disposons maintenant du volume mémoire de chaque procédure et nous pouvons calculer enfin celui du programme principal.

LEV	Place	Σ_t
(1)	6	6
(2)	4M+3N+5	4M+3N+11
(3)	N-1	4M+4N+10
(4)	8M+N+2	12M+5N+12
(5)	N	12M+6N+12
Call TREMAUX	4N+9	12M+9N+21
Call SORTING	2M+11N+4	6M+15N+14
(3)	4M-4N+5	8M-N+16
Call PLANAR	5M-3N+18	13M-4N+34
(4)	3M-3N+3	11M-4N+19
Call EMBED	4M-4N+12	15M-8N+31

A la lecture de ces résultats , on s'aperçoit que le programme occupe un espace mémoire maximal pendant l'exécution de la procédure "TREMAUX" , avec 12M+9N+21.

⇒ THEOREME: & (version C1) = 12M+9N+21 .

Remarque: M pouvant varier de N à 3N-6 , cet encombrement varie de même de 21N+21 à 45N-51 .

Exemple: Pour un graphe de 110 sommets et 324 arêtes , on obtient :

& (TREMAUX) = 4899 mots , & (PLANAR) = 3807 mots

& (SORTING) = 3608 — , & (EMBED) = 4011 — .

III/ ANALYSE EN "TEMPS"

1) GENERALITES

Dans cette analyse , nous allons assimiler le "temps" au "nombre d'opérations".

Ce programme n'ayant pas été écrit dans la seule optique des performances machine , il nous a semblé inutile de compter scrupuleusement chaque type d'opération.

Nous avons donc effectué un comptage grossier en "unité temps" (UT) , dont voici quelques règles :

Nature de l'opération		N^b d'UT
Adressage simple :	"X"	0
——— tableau :	"TAB(X)"	1
Affectation simple :	"A ← X"	1
Opération arithmétique:	"X +,-,x,÷ Y"	1

EXEMPLE: DUS(RTLEV) = FI(RTLEV) + FI(LB(LEV))

1 1 1

2

4

6 UT.

De même que nous avons défini une fonction espace "&" , définissons une fonction temps "T" , telle que T (TRUC) représente le nombre d' UT. nécessaire pour 1 exécution de la procédure "TRUC".

Introduisons également la fonction "T_G" , telle que T_G(TRUC) représente le nombre d' UT. nécessaire pour l'exécution complète du programme (c-à-d. pour le traitement d'un graphe G) , de la procédure "TRUC".

EXEMPLE: Si T (TRUC) = β et si la procédure "TRUC" est appelée systématiquement α fois pour le traitement d'un graphe , alors on peut écrire : $T_G(TRUC) = (\alpha.\beta)$.

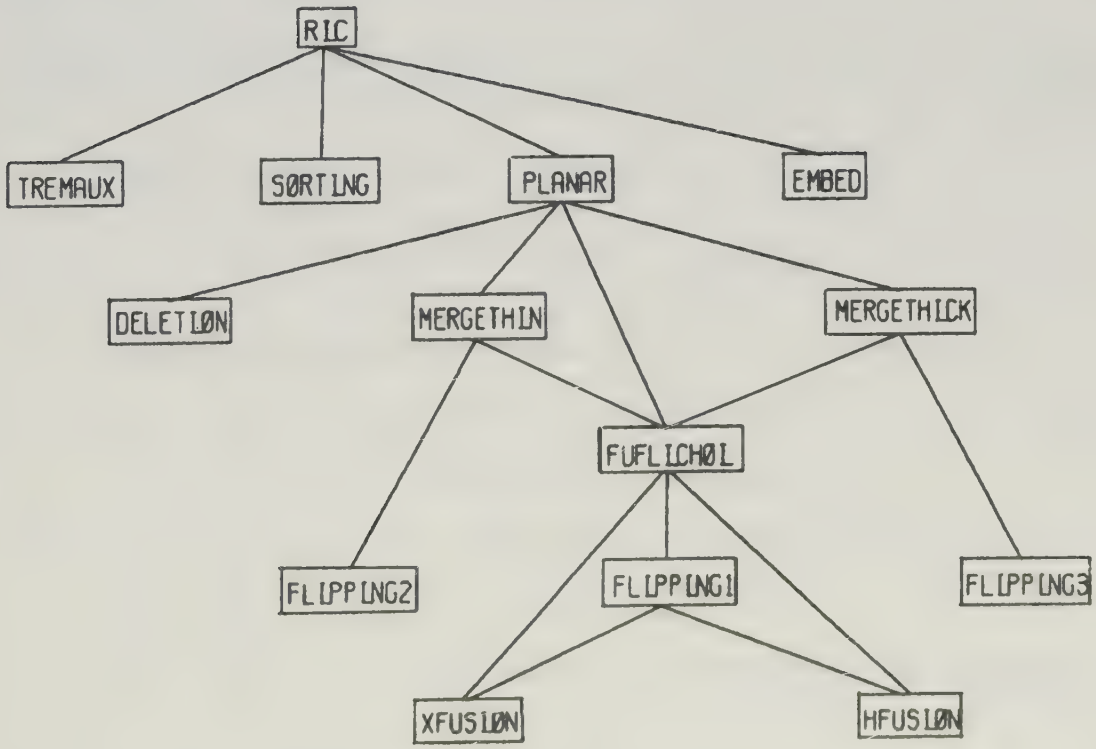


Schéma synoptique

Nous avons donc à calculer T (RIC) , = T_G(RIC). Nous n'obtiendrons d'ailleurs qu'une majoration de ce nombre.

Un rapide calcul montre que , sans compter les opérations d' Entrée/Sortie (GET ou PUT) : T (RIC) = 28M+4N+23

+ T (TREMAUX)

+ T (SORTING)

+ T (PLANAR)

+ T (EMBED).

2) CALCUL DE T (TREMAUX) ET T (SORTING)

Il est bien connu qu'une descente en profondeur (D.F.S.) dans un graphe est linéaire en $\mathcal{O}$(max(M,N)) si le graphe est représenté par une liste d'adjacence. [ref.H.S]

Nous rappelons que "SORTING" , utilisant le système de tri Radix-Sorting est également linéaire en $\mathcal{O}$(N). [ref.K.1]

Sans difficulté , on peut trouver pour notre programmation :

T (TREMAUX) $\leqslant$ 47M+39N-27

et T (SORTING) $\leqslant$ 28M+34N-6 .

3) CALCUL DE T (PLANAR)

C'est évidemment la partie la plus importante à analyser. Il va falloir nous occuper des neuf procédures appelées par "PLANAR" .

3-1) T_G(X ou H-FUSION)

Commençons par les deux plus simples : les sous-programmes de fusion. Trivialement on peut borner le nombre d'opération de ces fusions par une constante :

$$T\ (\text{XFUSION}) \leqslant 23 \quad \text{et} \quad T\ (\text{HFUSION}) \leqslant 21\ .$$

Analysons maintenant ces deux fusions globalement.

Considérons le fait suivant : "après chaque passage par XFUSION ou HFUSION , la hauteur de la TWUN (c-à-d. le nombre de double éléments) se trouve diminué de 1."

En effet , chaque fusion détruit systématiquement un plancher K de la TWUN.

NOTE: Nous rappelons que l'algorithme crée toujours (M-N+1) doubles éléments. (Lors de l'arrivée de chaque arc du coarbre , il crée un nouveau double élément.) Il est donc créé (M-N) planchers. Lorsque l'algorithme se termine , le graphe ayant été trouvé planaire , la TWUN est vide : tous les planchers ont donc été détruits. Les seuls endroits où peut disparaitre un plancher sont : les deux fusions , les deux merges et DELETION.

Soient S_f le nombre de doubles éléments Supprimés par les fusions ,

S_m ——————————————————— merges ,

S_d ——————————————————— DELETION ,

avec: $S_f + S_m + S_d = (M-N)$

■

En retenant que T (X ou H-FUSION) $\leqslant$ 23 , on obtient pour le graphe G :

$$T_G(\text{X ou H-FUSION}) \leqslant 23.S_f$$

Le sort des fusions est réglé , et dorénavant les "CALL fusion" vont disparaitre des sous-programmes à analyser.

Nous continuons l'analyse par les trois procédures de bascule.

3-2) T_G(FLIPPING1,2,3)

Définitions: - Soit "f" le nombre d'arcs (de coarbre) basculés lors d'un appel à FLIPPING1 , 2 ou 3 . ($f \geqslant 1$)

- Soit "F" le nombre de bascules élémentaires effectuées pour le traitement d'un graphe G. ($\Sigma f=F$)

Sans gros effort , on peut trouver:

$$T\ (\text{FLIPPING1}) \leqslant 48+22f$$
$$T\ (\text{FLIPPING2}) \leqslant 5+21f$$
$$\text{et}\ \ T\ (\text{FLIPPING3}) \leqslant 48+26f\ .$$

En majorant grossièrement , on peut dire qu'un appel à FLIPPING ne bascule qu'un seul arc : c'est le cas le moins 'rentable'.

Retenons donc que T (FLIPPING1,2,3) $\leqslant$ 48+26f , on obtient alors pour le graphe G :

$$T_G(\text{FLIPPING1,2,3}) \leqslant 74.F$$

Le sort des bascules est réglé et désormais , les appels à tous les mécanismes élémentaires de manipulation de la TWUN vont disparaître de notre analyse.

Nous arrivons à l'analyse de la procédure "FUFLICHOI".

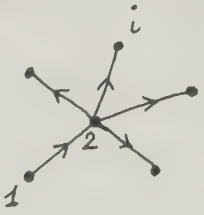
3-3) T_G(FUFLICHOI)

Propriété: Le nombre d'appels aux merges est au plus égal à N-3.
Cette borne pouvant être atteinte.

Preuve: L'exécution d'une procédure de merge 'soude' deux branches de l'arbre. Un arbre possèdant (N-1) arcs , le nombre de 'fourches' est trivialement borné par (N-2). Cette borne est atteinte pour un arbre en étoile.

Les deux seules orientations possibles pour notre arbre de Trémaux sont les suivantes :

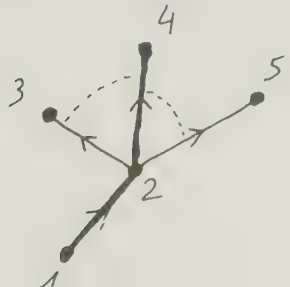
(1)

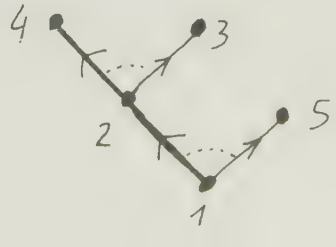
ou (2) 

Le graphe devant , par hypothèse , être simple , on ne peut construire aucun coarbre sur l'arbre de type (1).

Sur celui de type (2) , on obtiendra un tronc [1,2,i] et donc (N-3) fourches.

■

Exemples: 

ou 

Lorsque l'on place un arc dans la TWUN 'sur' un plancher KK , (le tout premier arc de coarbre traité par exemple) l'appel à "FUFLICHOI" (depuis "PLANAR") n'engendrera que l'exécution d'un seul test : l'arc étant seul dans cette partie de TWUN , le préordre est toujours respecté. Par contre , plus tard lorsqu'on sera amené à détruire ce plancher KK , l'appel à "FUFLICHOI" (depuis l'un des deux merges) sera quelconque.

Soit θ le nombre d'appels aux merges (donc de 'fourches').

La procédure "FUFLICHOI" est appelée (M-N+1) fois depuis le sous-programme "PLANAR" , dont θ fois qui n'engendrerons qu'un test , et donc θ fois depuis les deux procédures de merge. On obtient ainsi (M-N+1-θ) appels quelconques , plus θ tests , plus θ appels quelconques.

D'après la propriété vue précédemment , $\theta \leqslant$ N-3 .

Nous pouvons schématiser , pour notre analyse , la procédure "FUFLICHOI" de la façon suivante:

```
┌boucle
│ Si  'condition'
│ Alors  'entre autres on diminue la hauteur de
│         la TWUN de 1 , avec 10 UT.'
│ Sinon  RETURN avec 7 UT.
└repeat.
```

Le cas "Alors" , correspondant à une suppression de plancher par fusion , se produit S_f fois.

Le nombre de RETURN est borné par le nombre d'appels quelconques.

On obtient ainsi la majoration globale suivante:

$$T_G(\text{FUFLICHOI}) \leqslant (N-3) + 10.S_f + 7.(M-N+1)$$

$$\Rightarrow \quad T_G(\text{FUFLICHOI}) \leqslant 7M-6N+4+10S_f$$

3-4) T_G(élimination)

Pendant l'exécution du programme , les (M-N+1) arcs du coarbre sont éliminés de la TWUN : c'est ce temps que nous allons chercher à calculer.

Les arcs peuvent être:

- détruits (DELETE ; ERA=0) par la procédure "DELETION" ,
- effacés (ERASE) par
  - MERGETHIN et FLIPPING2 (ERA=1)
  - FLIPPING3 (ERA=2)

Les éliminations dues à FLIPPING2 et à FLIPPING3 sont déjà comptabilisées. Nous allons majorer le temps des éliminations car nous supposerons que tous les arcs sont éliminés soit par "DELETION" , soit par "MERGETHIN".

On trouve facilement que l'effacement d'un arc coûte 11 UT. par MERGETHIN , et que sa destruction par DELETION coûte 13 UT.

On obtient donc une majoration sur l'élimination globale des arcs :

$$T_G(\text{élimination}) \leqslant 13.(M-N+1)$$

3-5) FIN DE L'ANALYSE DE "PLANAR"

a) Il reste à analyser une partie des deux procédures de merge : la partie qui n'élimine pas les arcs. On trouve facilement :

$$T_p(\text{MERGETHIN}) \leqslant 28$$
$$\text{et } T_p(\text{MERGETHICK}) \leqslant 34 \qquad (p \text{ pour partiel})$$

D'après la propriété vue au paragraphe 3-3) , on majore le nombre de merges par (N-3) , et on obtient :

$$T_{pG}(\text{MERGE}) \leqslant 34.(N-3) + 21.S_m$$

b) Il reste également à analyser la partie de DELETION non vue dans l'élimination des arcs.

Le nombre d'appels à la procédure "DELETION" est fixe , et de (N-1). Nous pouvons schématiser cette neuvième procédure de façon analogue à ce que nous avons fait pour FUFLICHOI :

```
┌ boucle
│  Si  'condition'
│  Alors  'la hauteur de la TWUN diminue de 1 ,
│          avec un maximum de 20 UT.'
│  Sinon  RETURN avec un maximum de 19 UT.
└ repeat.
```

On obtient la majoration suivante :

$$T_{pG}(\text{DELETION}) \leqslant 20S_d+19.(N-1)$$

c) En récapitulant , nous allons enfin pouvoir calculer la valeur de T (PLANAR).

$$\begin{aligned} T\ (\text{PLANAR}) \leqslant\ & 18M+19N-5 + T_G(\text{X ou H-FUSION}) + T_G(\text{FLIPPING1,2,3}) \\ & + T_G(\text{FUFLICHOI}) + T_G(\text{élimination}) \\ & + T_{pG}(\text{MERGE}) + T_{pG}(\text{DELETION}) \ . \end{aligned}$$

Avec un peu de bonne volonté on obtient :

$$T\ (\text{PLANAR}) \leqslant\ 38M + 53N - 109 + 74F + 33S_f + 21S_m + 20S_d \ .$$

Majorons $(33S_f + 21S_m + 20S_d)$ par $33.(S_f+S_m+S_d)$, donc par $33.(M-N)$. On obtient finalement :

$$T\ (\text{PLANAR}) \leqslant\ 71M+20N-109+74F$$

4) CALCUL DE T (EMBED)

Il ne nous reste plus qu'à effectuer l'analyse de la 14$^{\text{ième}}$ et dernière procédure: EMBED.

On trouve sans aucun problème que :

$$T\ (\text{EMBED}) \leqslant\ 56.(M-N) + 65$$

5) CONCLUSIONS

La majoration du temps d'exécution du programme apparait en rassemblant tous les morceaux du puzzle (in english 'puzzle') :

⇒ THEOREME: T (version C1) $\leqslant$ 230M+41N-54+74F .

Vocabulaire: Nous appellerons "arc basculeur" l'arc responsable d'une bascule .

LEMME 1 : L'arc basculeur est toujours éliminé de la TWUN après l'arc basculé correspondant .

démonstration: évident car LV(basculeur) $<$ LV(basculé) .

LEMME 2 : L'arc basculeur interdit pour toujours une nouvelle bascule à l'arc basculé correspondant .

démonstration: l'arc basculé est bloqué , au niveau de la boite noire de validation de bascule , par le basculeur (même si celui-ci a été basculé) .

LEMME 3 : Si l'arbre de Trémaux est une chaine , alors aucun arc ne peut basculer plus d'une fois .

démonstration: LEMME 1 ⊕ LEMME 2 .

Soient l'arc α et l'arc β les deux hauts (Gauche et Droit) du double élément concerné par une bascule.

Soit par exemple l'arc β celui des deux hauts devant basculer.

LEMME 4 : La bascule de l'arc β interdit à l'arc α toute future bascule .

démonstration: Le basculeur de l'arc β bloque , pour des raisons de préordre , l'arc α .

THEOREME: $F \leqslant M - N$.

démonstration: LEMME 3 ⊕ LEMME 4 ⊕
cas Mergethin ⊕ cas Mergethick .

- Nous pouvons appliquer le Lemme 3 sur la partie 'dégagement du tronc' de chacun des deux merges d'après la propriété 2 (cf. p. 48) .

- L'arc de niveau K servant dans le "Call FUFLICHOI" se trouvant à la fin de chacun des deux merges peut être considéré comme un nouvel arc entrant dans la TWUN , et le Lemme 3 s'applique donc .

D'où la majoration finale :

⇒ THEOREME: T (version C1) $\leqslant 304\,M - 33\,N - 54{,}22$.

CHAPITRE IV

IMPLANTATION ET PROLONGEMENTS

I/ GENERALITES

La procédure "EMBED" , en fonction des paramètres "PRED" , "FI" , "DUS" et "ERA", calcule pour chaque arc de coarbre son Upper Side (US) , son Lower Side (LS) et son numéro de famille (LIB).

Le lien "PRED" , le Flipping Index (FI) , la Différence d'Upper Side (DUS) ainsi que l'indicateur d'élimination (ERA) ont été positionnés dans la procédure "PLANAR".

Le lien PRED(α) n'a pas été modifié depuis l'instant où l'arc α a été éliminé de la Twin Unit , soit par Erase lors d'un "MERGETHIN", soit par la procédure "DELETION".

FI(α) = 0 si les brins haut et bas de l'arc α sont du même côté , = 1 sinon. Remarquons que si un arc est basculé deux fois , son Flipping Index revient à sa valeur initiale.

US(α) = 0 si le brin haut de l'arc α est situé du côté gauche, = 1 s'il est situé du côté droit.

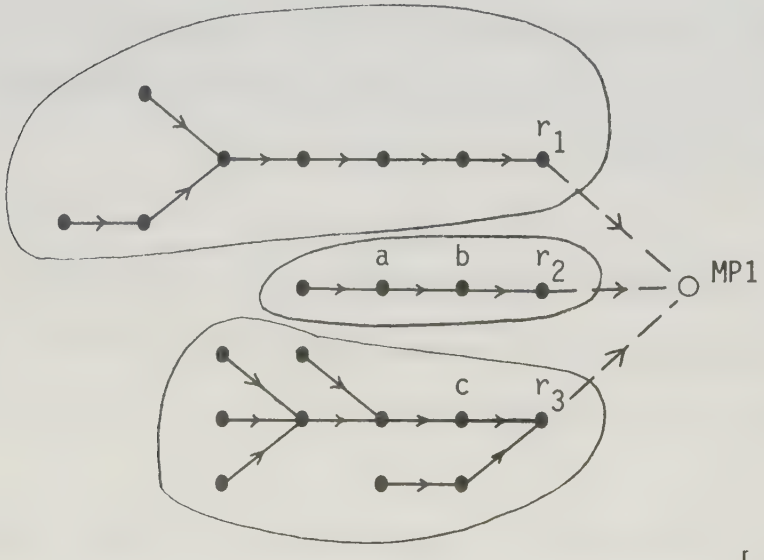
LS idem

β = PRED(α) ; DUS(α) = 0 si US(α) = US(β) , =1 sinon.

ERA(α) = 0 si l'arc α a été éliminé de la Twin Unit par la procédure "DELETION" , =1 sinon (c-à-d. par erase).

Lors de la disparition de chaque arc dans la Twin Unit , nous avons entre autres positionné le lien PRED .

Lorsque la procédure "PLANAR" est terminé , ce lien décrit un graphe orienté dont chaque sommet est un arc du coarbre.



[PRED(a) = b]

Nous allons construire ce graphe , mais en inversant l'orientation de chaque arc de façon à obtenir un arbre de racine MP1 .

II/ LA PROCEDURE "EMBED"

1) INTRODUCTION

Pour calculer la position de l'arc "a" , nous devons d'abord (à cause du DUS) calculer celle de l'arc "b". (cf. dessin)

Nous bâtirons donc le calcul des positions des arcs sur une descente en profondeur d'abord (D.F.S.) effectuée sur l'arbre de racine MP1. (cf. dessin)

Les liens $PRED(r_1)$, $PRED(r_2)$ et $PRED(r_3)$ n'ont pas de signification géométrique mais n'existent que comme effet de bord , pour faciliter l'écriture du programme, en manipulant une structure de donnée connexe. Tous les autres arcs en revanche symbolisent une relation géométrique.

Nous rappelons (cf. chapitre I , Interprétation géométrique) l'indépendance Gauche-Droite d'un double élément à l'intérieur de la Twin Unit. La position des arcs que nous calculons est donc toute relative.

Des arcs étant dans le même double élément sont reliés par le lien "PRED". Ce lien marque donc la solidarité existant entre deux arcs. C'est pourquoi l'arc "a" et l'arc "b" sont solidaires.

En 'repoussant' de proche en proche la relativité de position , nous arrivons en r_1 ou r_2 ou r_3 à faire un choix de position absolu , choix arbitraire.

Remarquons que les "r_i" sont les racines des composantes connexes du graphe.

Nous avons choisi la forme canonique suivante : " l'Upper Side de chaque racine r_i est situé à gauche." Nous tranchons ainsi dans cette perpétuelle relativité. C'est la position des arcs donnée par le programme.

Nous avons maintenant la signification du tableau "LIB" : il y a autant de familles d'arc indépendantes (au sens des positions G-D) que de racines r_i .

'Modulo' cette indépendance Gauche-Droite entre familles , nous pouvons obtenir différentes représentations planaires du graphe . Le programme donne une position pour chaque arc et un numéro de famille. Soit "l" le nombre de familles. Nous pouvons déduire 2^l configurations absolues de graphes.

2) ALGORITHME

Soit X = PRED(Y) , la position de l'arc X etant supposée connue.

Le problème est donc de calculer la position de l'arc Y , c-à-d. US(Y) et LS(Y).

Remarquons que LS(Y) = US(Y) + FI(Y) (*)

(*) : ce calcul est effectué modulo 2.

CALCUL

Si Y est une racine r_i

Alors
- US(Y) = 0 {choix arbitraire}
- LS(Y) = FI(Y) (*)

Sinon
- US(Y) = US(X) + DUS(Y) (*)
- Si Y a été éliminé par ERASE
  - Alors FI(Y) = FI(X) + DUS(Y) {restauration du FI}
- LS(Y) = US(Y) + FI(Y) (*)

Remarquons que les résultats fournis par ce programme (US et LS) ne suffisent pas pour dessiner le graphe sans ambiguïté : il manque le dessin de l'arbre (qui peut par quelques aménagements être récupéré lui aussi) .

Mais les ouvrages les plus courts
Sont toujours les meilleurs. En cela, j'ai pour guide
Tous les maîtres de l'art, et tiens qu'il faut laisser
Dans les plus beaux sujets quelque chose à penser...

JEAN DE LA FONTAINE
Fables, Livre X, 14, Les lapins.

~ BIBLIOGRAPHIE ~

[B.1] C. Berge , Graphes et Hypergraphes , Dunod , Paris 1973.

[B.L.] K.S. Booth et G.S. Leuker , Testing for Consecutives Ones Property , Interval Graphs , and Graph Planarity Using PQ-Tree Algorithms , Journal of Computer and System Sciences 13, 335-379 (1976).

[C.1] C.I.I. , Série 10070,série IRIS 80 , Langage PL1 sous SIRIS 7/ SIRIS 8 , Manuel d'utilisation. (1975)

[D.M.P.] G. Demoucron , Y. Malgrange et R. Pertuiset , Graphes Planaires , Revue Française de Recherche Opérationnelle Vol.30 , 33-47 (1964).

[F.R.1] H. de Fraysseix et P. Rosenstiehl , A Discriminatory Theorem of Kuratowski Subgraphs , Graph Theory Conference , February 10-13 1981 (edited by M.M. Syslò).

[F.R.2] H. de Fraysseix et P. Rosenstiehl , Système de référence de Trémaux d'une représentation plane d'un graphe planaire , Colloque International sur la Théorie des Graphes et la Combinatoire , Marseille-Luminy 15-19 Juin 1981 , Combinatorial Mathematics , Annals of Discrete Mathematics , 12 , North-Holland , 1982 .

[F.R.3] H. de Fraysseix et P. Rosenstiehl , A Depth-First-Search Characterization of Planarity , Proceeding of the 1981 Cambridge Combinatorial Conference (editor B. Bollobas).

[F.1] J.C. Fournier , Démonstration simple du théorème de Kuratowski et de sa forme duale , Discrete Mathematics 31 (1980) , 329-332 .

[G.1] A.J. Goldstein , An efficient and constructive algorithm for testing whether a graph can be embedded in a plane , Graph and Combinatorics Conference contact No-NONR 1858-21 , Office of Naval Research Logistics Proj., Dept. of Math. Princeton Univ. May 16-18 1963 .

[H.S.] E. Horowitz et S. Sahni , Fundamentals of Computer Algorithms, Pitman , London 1978 .

[H.T.1] J. Hopcroft et R. Tarjan , Planarity testing in V log V steps : Extended abstract. Proc. IFIP Cong. 1971 : Foundation of Information Processing , Ljubljana , Yugoslavia , Aug. 1971, North-Holland Pub. Co. , Amsterdam , 18-22

[H.T.2] J. Hopcroft et R. Tarjan , Efficient Planarity Testing , JACM Vol.21 , N°4 , October 1974 549-568 .

[K.1] D.E. Knuth , The Art of Computer Programming , Vol.3 , Addison-Wesley Publishing Company .

[L.E.C.] A. Lempel , S. Even et I. Cederbaum , An Algorithm for Planarity Testing of Graphs , Theory of Graphs : International Symposium , Rome , July 1966 , P. Rosenstiehl , Ed. Gordon and Breach , New York , 1967 , 215-232 .

[M.G.] P. Mei et N. Gibbs , A planarity algorithm based on the Kuratowski theorem , Proc. AFIPS 1970 , SJCC Vol. 36 , AFIP press Montvale , New Jersey 91-95 .

[R.N.D.] E.M. Reingold , J. Nievergelt et N. Deo , Combinatorial Algorithms , Prentice-Hall , Inc., Englewood Cliffs , New Jersey 1977 .

[S.1] A. Sache , La Théorie des Graphes , Presses Universitaires de France 1974 .